8-2020

Quantum and Classical Multilevel Algorithms for (Hyper)Graphs

Ruslan Shaydulin
Clemson University

Quantum and Classical Multilevel Algorithms for (Hyper)Graphs

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Science

---

by
Ruslan Shaydulin
August 2020

Accepted by:
Dr. Ilya Safro, Committee Chair
Dr. Yuri Alexeev
Dr. Amy Apon
Dr. Brian Dean
Dr. J. Cole Smith

Abstract

Combinatorial optimization problems on (hyper)graphs are ubiquitous in science and industry. Because many of these problems are NP-hard, development of sophisticated heuristics is of utmost importance for practical problems. In recent years, the emergence of Noisy Intermediate-Scale Quantum (NISQ) computers has opened up the opportunity to dramaticaly speedup combinatorial optimization. However, the adoption of NISQ devices is impeded by their severe limitations, both in terms of the number of qubits, as well as in their quality. NISQ devices are widely expected to have no more than hundreds to thousands of qubits with very limited error-correction, imposing a strict limit on the size and the structure of the problems that can be tackled directly. A natural solution to this issue is hybrid quantum-classical algorithms that combine a NISQ device with a classical machine with the goal of capturing "the best of both worlds".

Being motivated by lack of high quality optimization solvers for hypergraph partitioning, in this thesis, we begin by discussing classical multilevel approaches for this problem. We present a novel relaxation-based vertex similarity measure termed *algebraic distance for hypergraphs* and the coarsening schemes based on it. Extending the multilevel method to include quantum optimization routines, we present Quantum Local Search (QLS) – a hybrid iterative improvement approach that is inspired by the classical local search approaches. Next, we introduce the Multilevel Quantum Local Search (ML-QLS) that incorporates the quantum-enhanced iterative improvement scheme introduced in QLS within the multilevel framework, as well as several techniques to further understand and improve the effectiveness of Quantum Approximate Optimization Algorithm used throughout our work.

Dedication

To all the people who supported me on this journey.

Acknowledgments

First and foremost, I would like to thank my advisor, Ilya Safro, for his unwavering support. I would like to thank Yuri Alexeev for his mentorship and the doors he has opened for me.

I would like to thank my fellow ACS lab members at Clemson University, with whom I've shared a windowless office and many laughs: Justin Sybrandt, Ehsan Sadrfaridpour, Hayato Ushijima-Mwesigwa, Varsha Chauhan, Angelo Carrabba, Zirou Qiu, Ilya Tyagin and Joey Liu. I would like to thank my collaborators, many of whom have contributed directly to the results presented in this dissertation: Caleb Thomas, Christian Negre, Christopher Henry, Gianni Mossi, Jeffrey Larson, Jie Chen, Kaiwen Gui, Lukasz Cincio, Martin Suchara, Paige Rodeghero, Prasanna Balaprakash, Salvatore Mandra, Sami Khairy, Stuart Hadfield, Susan Mniszewski, Tad Hogg and Zain Saleem.

Lastly, I would like to thank Claire and my family for the love they've shown me. This work would not be possible without you.

Funding

Ruslan Shaydulin has been supported by the National Science Foundation under grant #1522751, the Defense Advanced Research Projects Agency (DARPA) and the Department of Energy Office of Science. High-performance computing resources at Clemson University were supported by National Science Foundation award MRI #1725573.

The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense, the Department of Energy or the U.S. Government.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Graphs and hypergraphs model a plethora of real-world phenomena, from the structure of human interactions to communication patterns in high-performance computing workloads. Understanding the properties of (hyper)graphs is instrumental in gaining an insight into many real-world problems. Unfortunately, many of the problems that arise in such analysis are extremely hard and finding exact solution is often computationally intractable. Therefore researchers resort to a variety of heuristical algorithms.

A crucial heuristical technique for the work described in this dissertation is local search. When a problem is too complex to tackle directly, it is intuitive to start with some initial solution (random or produced by a simple heuristic) and try to iteratively improve it. Variations of this idea of starting with some solution and iteratively searching its neighborhood have been successfully applied to a wide range of optimization problems, from training neural networks with stochastic gradient descent to solving general mathematical programming problems using simulated annealing. Local search is central to many algorithms described in this dissertation, most notably in the form of various local vertex-moving heuristics like KL-FM in hypergraph partitioning and quantum-accelerated large-neighborhood local search in Quantum Local Search.

A major limitation of local-search-based approaches is the limited quality of the initial solution. As most hard problems are non-convex and have multiple non-degenerate local optima, a bad initial solution can lead the iterative improvement scheme to get stuck in a local optimum. Many approaches were explored to tackle this, but the one we focus on in this dissertation is multilevel. A multilevel solver consists of three stages: coarsening, initial solution and uncoarsening.

During the coarsening stage, the problem is iteratively coarsened (reduced in size) with the goal of preserving the global structure. During the initial solution stage, the coarsened problem is solved, often exactly. During the uncoarsening, the solution is projected onto finer levels by reversing the coarsening operators and is iteratively refined using a local-search heuristic. The main advantage of this approach is that for each level in the multilevel hierarchy, the initial solution for the local search is "good" in a sense that it is optimal for the coarser problem. Assuming an optimal solution to the coarsest "initial" problem and coarsening that preserves the structure of interest, by induction the final solution to the finest, original problem will be optimal too. This intuition provides an insight into why multilevel methods have been so successful in tackling a variety of seemingly very different problems, from solving partial differential equations to image segmentation.

The above discussion provides a motivation for the work described in Chapter 2. Clearly, the quality of the initial solution projected from coarser levels is imperative for the success of the local-search-based refinement. That requires a good coarsening scheme capable of preserving the structure of the problem. Chapter 2 provides an in-depth discussion of advanced coarsening schemes for multilevel hypergraph partitioning. Joint work with Jie Chen on coarsening using a novel vertex similarity metric termed *algebraic distance for hypergraphs* is presented in Section 2.2. An extension of this approach to aggregative coarsening is presented in Section 2.3.

While a good initial solution is instrumental for the success of a local search heuristic, improving the local search procedure itself can yield great performance increases. Quantum Computation (QC) provides an enticing novel approach, with potential to revolutionize computation. Small quantum computers are becoming available on the cloud. These limited quantum computers are commonly referred to as Noisy Intermediate-Scale Quantum (NISQ) devices. NISQ-era quantum optimization techniques have the potential to provide speedups over classical state-of-the art optimization algorithms for certain problems. Two most prominent algorithms being explored are Quantum Approximate Optimization Algorithm (QAOA) on gate-based model quantum computers and Quantum Annealing (QA) on special-purpose quantum hardware. Chapter 3 provides an overview of quantum optimization techniques and a large scale numerical study of QAOA (Section 3.3) that presents some of the challenges associated with using QAOA. These challenges are partially addressed in a joint work with Jeffrey Larson on improving the performance of QAOA (Section 3.4). A further understanding of the mechanism of QAOA is gained through an application of graph theoretical approach to the analysis of structure of QAOA subspace. This approach and

its application to predicting QAOA performance with machine learning is discussed in Section 3.5. While both QAOA and QA can be understood in terms of local search, we focus on applying them as *subproblem* solvers within a larger iterative improvement scheme. Quantum optimization has the potential to explore larger neighborhoods of the current solution more efficiently than classical approaches. Chapter 4 presents Quantum Local Search (QLS, Section 4.1), a framework for community detection that uses quantum optimization techniques for improving the performance of a local-search approach to network community detection problem. This approach is extended to MultiLevel (ML-QLS) and graph partitioning problem in Section 4.2.

The following is the list of the technical results that appear in the dissertation:

1. R Shaydulin, J Chen, I Safro *Relaxation-Based Coarsening for Multilevel Hypergraph Partitioning* SIAM Multiscale Modeling & Simulation 17 (1), 482–506. Contribution: I developed and implemented the algorithm and was the primary contributor to writing all sections with the exception of Convergence Analysis (Sec. 4.2), which was contributed by J Chen.

2. R Shaydulin, I Safro *Aggregative Coarsening for Multilevel Hypergraph Partitioning* 17th International Symposium on Experimental Algorithms (SEA 2018), 103. Contribution: I developed and implemented the algorithm and was the primary contributor to writing all sections.

3. R Shaydulin, H Ushijima-Mwesigwa, I Safro, S Mniszewski, Y Alexeev *Community Detection Across Emerging Quantum Architectures* Proceedings of the 3rd International Workshop on Post Moore's Era Supercomputing. Contribution: I was the primary developer of the algorithm. I implemented the algorithm and was the primary contributor to writing all sections.

4. R Shaydulin, H Ushijima-Mwesigwa, I Safro, S Mniszewski, Y Alexeev *Network Community Detection on Small Quantum Computers* Advanced Quantum Technologies. Contribution: I was the primary developer of the algorithm. I implemented the algorithm and was the primary contributor to writing all sections.

5. R Shaydulin, I Safro, J Larson *Multistart Methods for Quantum Approximate Optimization* 2019 IEEE High Performance Extreme Computing Conference (HPEC). Contribution: I developed and implemented the algorithm and was the primary contributor to writing all sections.

6. H Ushijima-Mwesigwa, R Shaydulin, C Negre, S Mniszewski, Y Alexeev, I Safro *Multilevel Combinatorial Optimization Across Quantum Architectures* In submission. Contribution: I

contributed to the implementation of the algorithm. I ran the experiments in simulation and on IBM hardware and contributed to writing all sections. R Shaydulin and H Ushijima-Mwesigwa contributed equally to this work.

7. R Shaydulin, Y. Alexeev *Evaluating Quantum Approximate Optimization Algorithm: A Case Study* Proceedings of the 2nd International Workshop on Quantum Computing for Sustainable Computing (QCSC 2019) (in conjunction with 10th International Green and Sustainable Computing Conference (IGSC 2019)). Contribution: I performed all the experiments and was the primary contributor to writing all sections.

8. R Shaydulin, S Hadfield, T Hogg, G Mossi, J Sybrandt, I Safro *Symmetry can Predict QAOA Performance* In preparation. Contribution: I was the primary contributor to the proofs. I carried out the experiments and was the primary contributor to writing all sections.

Chapter 2

Advanced Coarsening Schemes for Multilevel Hypergraph Partitioning

Hypergraphs are generalizations of graphs. Both graphs and hypergraphs are ordered pairs of sets (V, E), where V is the vertex set and E is the set of (hyper)edges such that each $e \in E$ is a subset of vertices. The difference is that in a graph, the cardinality of each edge is exactly two, whereas in a hypergraph, a hyperedge can contain an arbitrary number of vertices. The hypergraph partitioning (HGP) problem is therefore a generalization of the graph partitioning (GP) problem. In GP the goal is to split the set of vertices into multiple sets (usually called partitions) of similar sizes, such that a cut metric is minimized. Here, a cut defines a set of (hyper)edges spanning more than one partition. In the HGP generalization, the hyperedges can possibly span more than two partitions. There exist several versions of minimization objectives, constraints, and cut metrics in both GP and HGP [31, 45]. Hypergraph partitioning has many applications, including VLSI design [130, 13], parallel matrix multiplication [47], classification [266], cluster ensembling [235], and combinatorial scientific computing [171], among others [31, 184].

2.0.1 Multilevel method

The multilevel method for graph partitioning was initially introduced to speed up existing algorithms [24], but was quickly recognized as a good way to improve the quality of partitioning [129, 110]. It is used as a global suboptimal heuristic framework [45], in which other heuristics are incorporated at different stages. These three stages are coarsening, initial partitioning, and uncoarsening. A multilevel hypergraph partitioning of a hypergraph constructed from `LPnetlib/lp_scfxm2` matrix is presented in Figure 2.1.

During the coarsening stage a hypergraph $H = (V, E)$ is approximated via a series of successively smaller hypergraphs $H^i = (V^i, E^i)$, $1 \leq i \leq l$, where l is the number of levels in the hierarchy. The superscript denotes the number of the corresponding level for hypergraphs, nodes, and edges, respectively. Each next-coarse hypergraph is constructed by merging or aggregating vertices in the previous one according to some heuristic: $v_k^i = \{v_{k_1}^{i-1}, ..., v_{k_j}^{i-1}\}$. That is, a vertex v_k^i in the coarse hypergraph G^i at the ith level is created by grouping a set of vertices $v_{k_1}^{i-1}, ..., v_{k_j}^{i-1}$ from the finer hypergraph H^{i-1}. Vertices can be grouped by using different criteria, with the aim of interpolating solution from coarse level nodes to the corresponding fine level nodes with minimum loss of solution quality. In the case of pairwise grouping of vertices, such a coarsening is referred to as matching. The weight of the new coarse vertex is equal to the sum of weights of the merged vertices:

$$w(v_k^i) = w(v_{k_1}^{i-1}) + ... + w(v_{k_j}^{i-1}).$$

A coarse vertex is contained in all hyperedges that contain the merged vertices. Hyperedges of cardinality one are discarded. Coarsening terminates when the size of the hypergraph is below a certain threshold or when a solution is easy to compute.

In the partitioning stage, the coarsest hypergraph H^l is partitioned using exact or approximate solver. In many existing implementations, the solver is a local search heuristic. This partitioning is anticipated to approximate the *global* solution in the sense that it incorporates the global structure of the hypergraph. In some cases, when H^l is sufficiently small, an exact solution can be computed.

The uncoarsening stage consists of two steps, namely, interpolation and refinement. During uncoarsening, the partitioning from the coarse hypergraph H^{i+1} is projected onto the fine H^i (interpolation) and refined using a local search heuristic such as Kernighan-Lin (KL) [132] or Fiduccia-

Mattheyses (FM) [79] (refinement). This retains the global information of the partitioning of the coarse hypergraph while optimizing it locally. Typically, solving a local search subproblem only improves the global solution at the same level.

2.1 Related work

Because HGP is NP-complete [84], many heuristics and approximations have been developed. The most common practical approach to HGP is the multilevel framework. This section begins with a brief description of non-multilevel techniques, followed by multilevel ones.

2.1.1 Spectral methods

An important family of non-multilevel techniques is spectral methods. It is necessary to point out that while they can be used as standalone methods, they are also often used within the multilevel framework. As we discussed in Section 2.0.1, the multilevel method for combinatorial optimization problems is a heuristic that incorporates other heuristics as well, such as different similarity concepts and iterative refinement techniques. Spectral hypergraph partitioning generalizes spectral graph partitioning methods to hypergraphs. It usually utilizes the spectral properties of the adjacency matrix. Two main approaches are identified.

The first one is to construct a graph from the hypergraph [102] and then apply spectral graph partitioning methods that are more well-developed [177, 80]. Two of the most common approaches are star and clique expansions. In the case of clique expansion, a hyperedge is replaced by a set of edges that form a complete subgraph for the vertices in the hyperedge. In the case of star expansion, a hyperedge is replaced by a new vertex, which is connected by new edges to all vertices previously contained in the hyperedge.

This approach suffers from an obvious loss of information: when a hyperedge is expanded (i.e., replaced by a clique or a star), its vertices are connected by a number of edges. The information that they are equal members of a hyperedge is lost. Ihler et al. show [118] that even for bi-partitioning there exists no min-cut graph model of a hypergraph. That is, one cannot create a graph whose edge cut is equal to the hyperedge cut in the original hypergraph, if negative weights are not allowed [118]. Finally, the hypergraph-to-clique conversion greatly increases the size of the problem. Nevertheless, we point out that despite these limitations, good practical results can still be obtained by using

graph models of a hypergraph.

The second approach is to build hypergraph Laplacian and to study its properties, bypassing the graph representation. This can be done in various ways. Bolla defines an unweighted hypergraph Laplacian matrix and shows the link between its spectral properties and the hypergraph cut [33]. Zhou et al. define a Laplacian matrix and show a way to use it for k-way partitioning [266]. Hu et al. argue that Laplacian tensors naturally extend the graph Laplacian matrices to hypergraphs. They describe a Laplacian tensor for an even uniform hypergraph and define algebraic connectivity for it [114]. Chan et al. define a Laplacian operator induced by a stochastic diffusion process on the hypergraph and generalize Cheeger's inequality for it [50]. However, these recent advances of the spectral approaches, while promising, are not yet well developed for large-scale instances.

2.1.2 Multilevel methods

Most state-of-the-art hypergraph partitioners (such as hMetis2 [130], PaToH [46], Zoltan [67] and Mondriaan [248], to name a few) use a multilevel approach inspired by simplified multigrid and the principles of multiscale computing.

In the coarsening stage of the V-cycle, most hypergraph partitioners use, with some variations, a heuristic that greedily aggregates neighboring vertices, with some preference based on a similarity metric. These similarity metrics are local and usually very simple. The metric used by Mondriaan [248], hMetis2 [130] and Zoltan [67] is *inner product matching*. The inner product of two vertices is defined as the Euclidean inner product of their hyperedge incidence vectors [67]. In other words, the inner product of two vertices is the number of hyperedges they have in common or, in the weighted case, total weight of those hyperedges. PaToH uses a different metric as a default option, called *absorption matching*:

$$\mathrm{am}(u,v) = \sum_{e=(u,v)\in E} \frac{1}{|e|-1},$$

where $|e|$ is the cardinality of an edge e. However, there are scenarios where these simple solutions, while computationally efficient and easy to implement, are not very effective and can be improved [52]. This is the major reason for us to revise the coarsening strategy for hypergraphs. In the refinement step, all of the aforementioned partitioners use some variation of Fiduccia-Mattheyses [79] or Kernighan-Lin [132] including their advanced efficient implementation introduced in [9].

However, there is relatively little research on how to improve the coarsening of a hypergraph.

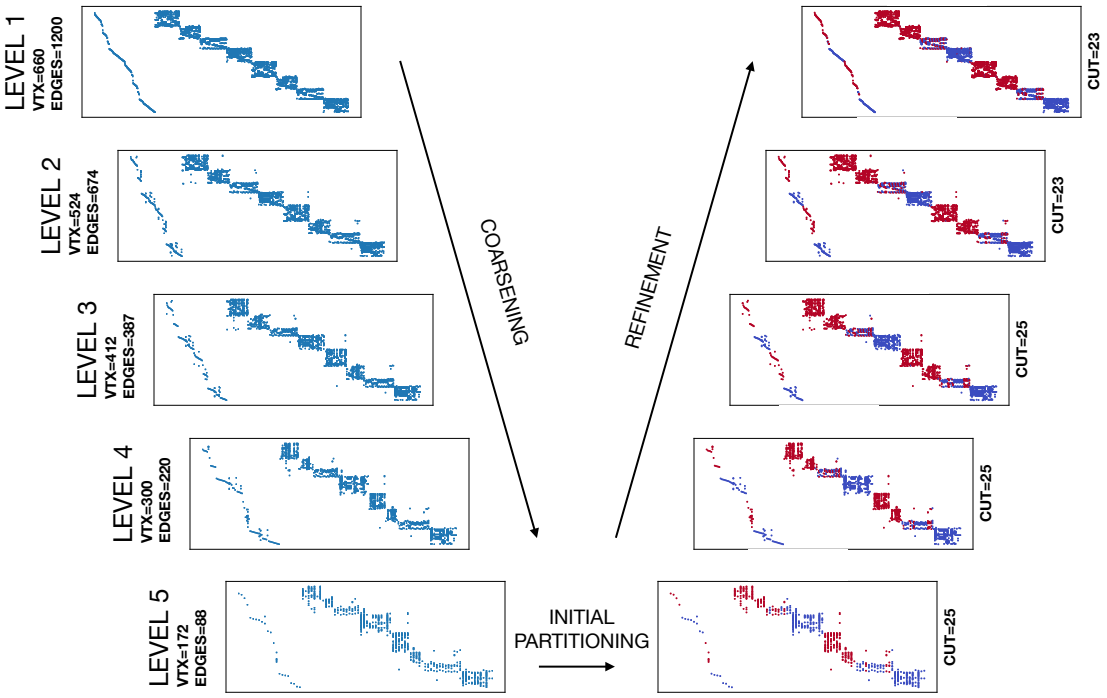


Figure 2.1: Multilevel partitioning of a hypergraph constructed from `LPnetlib/lp_scfxm2` matrix from SuiteSparse Matrix Collection [64] using row-net model: each column becomes a vertex and each row becomes a hyperedge. On the left side of the 11V" the hypergraph (represented here as the sparsity pattern of the underlying matrix) is iteratively coarsened. At the bottom of the "V" the hypergraph is partitioned into two parts. This is represented by coloring the columns corresponding to vertices from one part into blue and another into red. On the right side of the "V" the hypergraph is uncoarsened and the partitioning is refined.

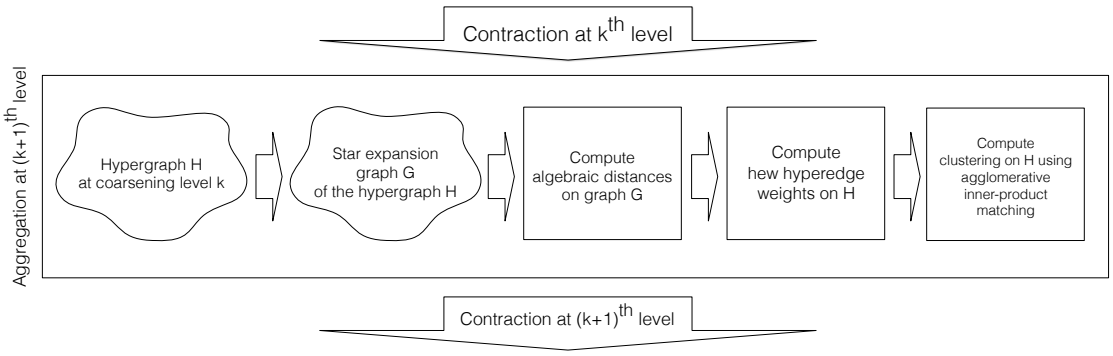


Figure 2.2: Outline of the algorithm used to aggregate vertices at kth coarsening level.

Existing research was motivated mainly by the intuition that a decision made earlier in the coarsening stage has substantial influence on the quality of the final cut: any error or wrong decision would be propagated all the way down the V-cycle and accumulate. Some methods that improve the coarsening include a use of rough set theory [146], as well as some variations on the greedy heavy matching scheme. A very promising but unfinished attempt to generalize HGP coarsening using algebraic multigrid was published in Sandia Labs Summer Reports [44]. Another extension of a multilevel method, namely, the n-level recursive bisection, was introduced in [218]. For a more extensive review of hypergraph partitioning algorithms the reader is referred to [244].

2.2 Algebraic Distance on Hypergraphs

In this section, we introduce a new distance measure that plays a crucial role in improving the quality of coarsening. To demonstrate its effectiveness, we use Zoltan [67] as the baseline solver.

The outline of our approach is as follows. At each level of coarsening, we compute new weights for the hyperedges. These weights are passed to Zoltan's coarsening subroutine, allowing it to use this additional information for making matching decisions. After the matching is computed, the weights are set back to the original hyperedge weights and the multilevel algorithm continues. In other words, we leverage the hyperedge weights to pass information on the structure of the hypergraph, derived by our algorithm, to the HGP solver's coarsening scheme. We refer to these weights as *algebraic weights*. The outline is shown in Figure 2.2.

Discussion of this algorithm would not be complete without a brief description of the coarsening scheme used by Zoltan. Zoltan uses an agglomerative matching technique known as inner-

product matching [48], or called heavy-connectivity clustering in PaToH [47]. In this technique, the vertices are visited in random order. If the visited vertex v is unmatched, an adjacent vertex u with the highest *connectivity* is selected and the current vertex is added to its cluster C_u. The *connectivity* is defined as $N_{v,C_u}/W_{v,C_u}$, where N_{v,C_u} is the total weight of the edges connecting v with the vertices in the cluster C_u, and W_{v,C_u} is the total weight of the vertices in the candidate cluster [47].

2.2.1 Algorithm

Recall that the hypergraph is denoted as $H = (V, E)$. We call the star expansion graph G. In general, we use primed variables when referring to the elements of G and non-primed for the elements of H. At each coarsening level the following happens:

1. Build the star expansion graph $G = (V', E')$ of H [237]. Each hyperedge is replaced by a new vertex, which is connected to every vertex contained in the original hyperedge, i.e., $V' = V \cup E$ and $E' = \{(v, h) \mid v \in h, h \in E\}$ [237]. This way, each hyperedge is replaced by a "star" in the bipartite graph G, with all vertices $v \in V$ in one part and hyperedges $h \in E$ in another (hence the new set of vertices is $V' = V \cup E$). The weights of the vertices stay the same. The vertices introduced to replace hyperedges are assigned the weights of the corresponding hyperedges they represent, divided by the number of vertices in that hyperedge; i.e.,

$$w'(v') = w(v), \quad \forall v \in V \subset V', \text{ and}$$

$$w'(h') = \frac{w(h)}{|h|}, \quad \forall h \in E \subset V'. \tag{2.1}$$

   The edges $e' \in E'$ of the star expansion G are left unweighted (we can assume they all have weight 1). We refer to the vector of weights in the graph G as $\mathbf{w'} \in \mathbb{R}^{|V'|}$ and that in the hypergraph H as $\mathbf{w} \in \mathbb{R}^{|V|+|E|}$, to distinguish the two cases. Note that since $V' = V \cup E$, these vectors are of the same size.

2. Compute algebraic coordinates of the vertices in star expansion graph G by using Jacobi over-relaxation (JOR, a stationary iterative relaxation), beginning with random initialization. The iterative process is repeated for several random initial vectors.

For each random vector, all coordinates are initialized with random values from the uniform distribution on $(-0.5, 0.5)$. We denote the algebraic coordinates as a 2-dimensional array X, where $x[r][v']$ is the algebraic coordinate for the vertex $v' \in V' = V \cup E$ and the rth random initial vector. Then, for a certain number of iterations (denoted in pseudocode as *num_iter*), JOR is performed on all $v' \in G$ as follows. The vertices are visited in random order. After each iteration, the algebraic coordinates are rescaled such that the smallest algebraic coordinate is equal to -0.5 and the largest to 0.5. The iteration scheme can be written as follows

$$x^*[r][v'] = \frac{\sum_{\forall u' \in V' : (u', v') \in E'} w'(u') \cdot x^{i-1}[r][u']}{\sum_{\forall u' \in V' : (u', v') \in E'} w'(u')} \tag{2.2}$$

$$x^{**}[r][v'] = \omega \cdot x^*[r][v'] + (1 - \omega) x^{i-1}[r][u'],$$

where ω is a relaxation factor. It is used in the same way as in Successive overrelaxation [43], to make the convergence more stable. Weighting, rescaling and the denominator in Equation 2.2 are introduced in part to prevent algebraic coordinates from converging to machine precision, i.e., becoming so close that they are no longer distinguishable. In our experiments, nondistinguishable coordinates tend to occur on hypergraphs whose hyperedges have high cardinality, since vertices contained in these hyperedges very quickly "pull" each other together.

Denote the smallest ("leftmost" on the real line) algebraic coordinate before rescaling as l and the largest ("rightmost") as r. Then, the rescaling is done as follows:

$$x^i[r][v'] = \frac{x^{**}[r][v'] - l}{r - l} - 0.5.$$

3. Define the algebraic weight of a hyperedge $h \in E$ as one over the algebraic distance between two farthest apart vertices in h, maximized over all random vectors:

$$alg\_weight(h) = \frac{1}{\max\limits_{\forall r} \max\limits_{u,v \in h} |x[r][v] - x[r][u]|}.$$

Compute the final weights $\tilde{\mathbf{w}}$ to be passed to Zoltan from the original hyperedge weights $\mathbf{w}$ as

follows

$$\tilde{w}(h) = w(h) \cdot \frac{alg\_weight(h)}{\sum_{h^* \in E} alg\_weight(h^*)/|E|}.$$

Hyperedge weights are multiplied by the ratio between the computed algebraic distance for this hyperedge and the average algebraic weight for all hyperedges. Note that here $w(h)$ is the weight of the original hyperedge, before the star expansion.

4. Pass these new weights $\tilde{\mathbf{w}}$ to Zoltan's [67] agglomerative inner product matching. After the matching, all weights are reset back to the ones before the star expansion (i.e., back to $\mathbf{w}$) and the multilevel process continues.

The pseudocode for computing the algebraic weights (steps 1–3) is presented in Algorithm 1.

ALGORITHM 1: Computing algebraic weights

Input : Relaxation factor ω, number of random vectors R , number of iterations num_iter
Output: algebraic weights alg_weight

for $r = 1 : R$ **do**
  Randomly initialize $x[r]$;
  for $k = 1 : num\_iter$ **do**
    `// Perform iteration sweep over all vertices`
    for $v' \in V'$ **do**
      $x^*[r][v'] = \frac{\sum_{\forall u' \in V' : (u', v') \in E'} w'(u') \cdot x^{i-1}[r][u']}{\sum_{\forall u' \in V' : (u', v') \in E'} w'(u')}$;
      $x^{**}[r][v'] = \omega \cdot x^*[r][v'] + (1 - \omega) x^{i-1}[r][u']$;
    end
    for $v \in V'$ **do**
      `// Rescale`
      $l = \min_{u \in V'} x[r][u]$;
      $r = \max_{u \in V'} x[r][u]$;
      $x^i[r][v'] = \frac{x^{**}[r][v'] - l}{r - l} - 0.5$;
    end
  end
end
for h *in* E **do**
  `// Compute algebraic weights for hyperedges`
  $alg\_weight(h) = 1 \,/\, \max_{\forall r} \max_{u,v \in h} |x[r][v] - x[r][u]|$;
end

2.2.2 Convergence analysis

Algorithm 1 is an iterative process that computes $x[r][v']$ for all vertices $v' \in V'$ and random initial vector numbers r. To analyze the convergence of this process and understand the properties of the computed algebraic weights, we need additional notation. Let $x^{(i)} \in \mathbb{R}^{|V'|}$ denote the ith iterate of the vector $x[r]$ for the rth random initial vector. Since the same iterative process is performed for all vectors of random initial coordinates, we will perform the analysis on only one (arbitrary) vector and hence omit the dependence on r.

Let $A \in \mathbb{R}^{|E|\times|V|}$ be the incidence matrix of the hypergraph; that is, $A_{ij} = 1$ if vertex j belongs to the hyperedge i and 0 otherwise. Let $S^v \in \mathbb{R}^{|V|\times|V|}$ and $S^h \in \mathbb{R}^{|E|\times|E|}$ be diagonal matrices such that

$$S^v_{jj} = w(v_j) \quad \text{and} \quad S^h_{ii} = \frac{w(h_i)}{|h_i|},$$

where $|h_i|$ denotes the cardinality of the ith hyperedge (same as in Equation 2.1). Define the following matrix

$$W = \begin{bmatrix} 0 & A^T S^h \\ AS^v & 0 \end{bmatrix}$$

and let D be the diagonal matrix with elements $D_{jj} = \sum_i W_{ij}$. For convenience of analysis, we decompose D as

$$D = \begin{bmatrix} D^v & 0 \\ 0 & D^h \end{bmatrix}$$

with $D^v \in \mathbb{R}^{|V|\times|V|}$ and $D^h \in \mathbb{R}^{|E|\times|E|}$, such that the block sizes of D are compatible with those of W. Note that W is asymmetric.

It is not hard to see that Algorithm 1 computes the following update:

$$x^{(i)} = \frac{1}{r-l}\Big[\underbrace{\omega D^{-1}Wx^{(i-1)} + (1-\omega)x^{(i-1)}}_{x^{*(i-1)}}\Big] - \frac{r+l}{2(r-l)}\mathbf{1}$$

where $\mathbf{1}$ is the vector of all ones, and r and l are the maximum and the minimum of the elements in $x^{*(i-1)}$, respectively (see Algorithm 1). Then, we simplify the update formula as

$$x^{(i)} = \alpha^{(i-1)}Hx^{(i-1)} + \beta^{(i-1)}\mathbf{1}, \tag{2.3}$$

where

$$H = \omega D^{-1}W + (1-\omega)I, \quad \alpha^{(i-1)} = \frac{1}{r-l}, \quad \text{and} \quad \beta^{(i-1)} = -\frac{r+l}{2(r-l)}.$$

The convergence of the iteration depends on the properties of the iteration matrix H. We first note that the matrix $D^{-1}W$ is diagonalizable with real eigenvalues.

Theorem 2.2.1. *Let $(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}$ admit the following singular value decomposition:*

$$(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2} = \sum_{i=1}^{r} \sigma_i u_i z_i^T.$$

Then, $D^{-1}W$ has a zero eigenvalue of multiplicity $|V| + |E| - 2r = |V'| - 2r$. Moreover, for the nonzero eigenvalues,

$$D^{-1}W \begin{bmatrix} (S^v)^{-1/2}(D^v)^{-1/2}z_i \\ (S^h)^{-1/2}(D^h)^{-1/2}u_i \end{bmatrix} = \sigma_i \begin{bmatrix} (S^v)^{-1/2}(D^v)^{-1/2}z_i \\ (S^h)^{-1/2}(D^h)^{-1/2}u_i \end{bmatrix} \tag{2.4}$$

and

$$D^{-1}W \begin{bmatrix} -(S^v)^{-1/2}(D^v)^{-1/2}z_i \\ (S^h)^{-1/2}(D^h)^{-1/2}u_i \end{bmatrix} = -\sigma_i \begin{bmatrix} -(S^v)^{-1/2}(D^v)^{-1/2}z_i \\ (S^h)^{-1/2}(D^h)^{-1/2}u_i \end{bmatrix}, \quad i = 1, \dots r. \tag{2.5}$$

Proof. The identities 2.4 and 2.5 are straightforward to verify based on the singular value decomposition. On the other hand, if λ is an eigenvalue of $D^{-1}W$ with corresponding eigenvector $\left[\begin{smallmatrix} c \\ d \end{smallmatrix}\right]$, then,

$$(D^v)^{-1}A^T S^h d = \lambda c \quad \text{and} \quad (D^h)^{-1}AS^v c = \lambda d,$$

which implies that

$$[(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}]^T[(D^h)^{1/2}(S^h)^{1/2}d] = \lambda[(D^v)^{1/2}(S^v)^{1/2}c],$$

and

$$[(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}][(D^v)^{1/2}(S^v)^{1/2}c] = \lambda[(D^h)^{1/2}(S^h)^{1/2}d].$$

In form, these two equalities define all nonzero singular values $|\lambda|$ of $(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}$. Thus, if λ is not equal to any of the $\pm\sigma_i$'s, then λ must be zero. □

Second, note that the diagonal of the matrix $D^{-1}W$ is zero and each row is nonnegative, summing to one. Hence, by Gershgorin's circle theorem, the eigenvalues of $D^{-1}W$ must lie between -1 and 1. Indeed, the two ends of this interval are attainable.

Theorem 2.2.2. *The spectral radius of $D^{-1}W$ is 1. In particular, there exists a pair of eigenvalues ± 1, and $\mathbf{1}$ is an eigenvector associated with an eigenvalue $+1$.*

Proof. By the definition of W and D, we have $D^{-1}W\mathbf{1} = \mathbf{1}$. Therefore, $D^{-1}W$ has an eigenvalue $+1$ with eigenvector $\mathbf{1}$. From Theorem 2.2.1 we know that $D^{-1}W$ must also have an eigenvalue -1. Then, because all the eigenvalues of $D^{-1}W$ lie between -1 and $+1$ by the Gershgorin's circle theorem, its spectral radius is 1. □

Third, recall that the star expansion graph is a bipartite graph, with one part containing the vertices of the original hypergraph and the other part the hyperedges. Rather than defining vertex weights as in Equation 2.1 and leaving the edges unweighted, alternatively, one may view W as the weighted adjacency matrix of a graph with exactly the same vertex-edge structure, but the edges are weighted according to W and the vertices are left unweighted. Then, the block and the sparsity structure of W indicates that the directed edges of this bipartite graph (in the alternative view) always come in pairs with opposite directions. Hence, we may ignore the directions and treat the graph undirected, if at all convenient for analyzing graph connectivity. Then, the directed version of the graph is strongly connected if and only if the undirected version is connected, although the edge weights are asymmetric in the directed version. Consequently, the number of strongly connected components of the directed version is equal to the number of connected components of the undirected version. Such a view allows the application of the Perron–Frobenius theorem for exploring the extreme eigenvalues of $D^{-1}W$.

Theorem 2.2.3. *The eigenvalues of $D^{-1}W$ equal to $+1$ are all simple. The number of such eigenvalues is equal to the number of strongly connected components of the directed version of the bipartite graph (or equivalently, the number of connected components of the undirected version of the graph).*

Proof. If the directed graph is strongly connected, then $D^{-1}W$ is irreducible. Hence, by the Perron–Frobenius theorem, there exists an eigenvalue equal to the spectral radius 1. This eigenvalue is simple and is unique. If the graph has C strongly connected components, then $D^{-1}W$ may be symmetrically permuted into a $C \times C$ block-diagonal matrix, where each block corresponds to one

strongly connected component. In such a case, $D^{-1}W$ has C eigenvalues equal to 1, all of which are simple. $\square$

The above three theorems reveal the beautiful symmetry of the eigenvalues of $D^{-1}W$. They are real, come in pairs, and straddle around zero (except those being exactly zero). The number of nonzero pairs is equal to the rank of the incidence matrix A. The eigenvalues all lie inside $[-1, 1]$. Moreover, there exists (at least) one pair attaining exactly ± 1, and the number of such pairs is equal to the number of connected components of the star expansion graph when viewed as undirected.

With this knowledge, we see that there is a one-to-one correspondence between the eigenpairs of the iteration matrix H and those of $D^{-1}W$. Specifically, denote by (μ_i, ϕ_i) an eigenpair of H. Then,

$$H\phi_i = \mu_i\phi_i \quad \Leftrightarrow \quad D^{-1}W\phi_i = \frac{\mu_i - 1 + \omega}{\omega}\phi_i, \quad 0 < \omega < 1.$$

Because the eigenvalues of $D^{-1}W$ are symmetric around 0 with range $[-1, 1]$, the eigenvalues μ_i of H are symmetric around $1 - \omega$ with range $[1 - 2\omega, 1]$. Because ω is strictly less than 1, the largest eigenvalue of H is 1, which is equal to the largest eigenvalue in magnitude. The multiplicity of this eigenvalue is equal to the number of connected components of the undirected version of the star expansion graph. For simplicity of analysis, we will assume from now on that the graph is connected. Then, we have an ordering of the eigenvalues according to their magnitude:

$$1 = \mu_1 > |\mu_2| \geq |\mu_3| \geq \cdots \geq |\mu_{|V'|}|,$$

with $\phi_1 = \mathbf{1}$ being an (unnormalized) eigenvector associated with μ_1. Note the use of the strictly-greater-than sign $>$ and the greater-than-or-equal-to sign $\geq$. In particular, the second eigenvalue, in magnitude, must be strictly less than 1. In what follows, we will present a result that relates the difference between the elements of $x^{(k)}$ to that between the corresponding elements of some vector in the eigensubspace spanned by one or a few eigenvectors, including ϕ_2. This eigensubspace depends on how many eigenvalues are equal to $|\mu_2|$ in magnitude. If only one, that is, $|\mu_2| > |\mu_3|$, then the eigensubspace is spanned by only ϕ_2. However, if more than one, then let us assume that $|\mu_2| = |\mu_3| = \cdots = |\mu_t| > |\mu_{t+1}|$. Such a case includes two subcases:

case 1: $\mu_2 = \mu_3 = \cdots = \mu_t$; and

case 2: $\mu_2, \mu_3, \ldots, \mu_t$ are not all equal.

In each subcase, the vector is some linear combination of $\phi_2, \ldots, \phi_t$.

Theorem 2.2.4. *Assume that the undirected version of the star expansion graph is connected. Let the initial iterate $x^{(0)}$ be expanded in the eigen-basis of H as*

$$x^{(0)} = a_1\phi_1 + a_2\phi_2 + \cdots + a_{|V'|}\phi_{|V'|}.$$

(i) If $\mu_2 = \mu_3 = \cdots = \mu_t$ and $|\mu_t| > |\mu_{t+1}|$ for some $t \geq 2$, and if $a_2, \ldots, a_t$ are not all zero, then for any pair i, j,

$$\lim_{k\to\infty} \frac{(x^{(k)})_i - (x^{(k)})_j}{\alpha^{(0)}\alpha^{(1)}\cdots\alpha^{(k-1)}\mu_2^k} = \xi_i - \xi_j,$$

where

$$\xi = a_2\phi_2 + \cdots + a_t\phi_t.$$

(ii) If $|\mu_2| = |\mu_3| = \cdots = |\mu_t| > |\mu_{t+1}|$ for some $t \geq 3$ where $\mu_2, \mu_3, \ldots, \mu_t$ are not all equal, and if $a_2, \ldots, a_t$ are not all zero, then for any pair i, j,

$$\lim_{k\to\infty} \frac{(x^{(2k+p)})_i - (x^{(2k+p)})_j}{\alpha^{(0)}\alpha^{(1)}\cdots\alpha^{(2k+p-1)}\mu_2^{2k+p}} = (\eta_p)_i - (\eta_p)_j,$$

where

$$\eta_p = a_2\phi_2 + a_3(\mu_3/\mu_2)^p\phi_3 + \cdots + a_t(\mu_t/\mu_2)^p\phi_t, \quad p = 0, 1.$$

Proof. For notational convenience, let $n = |V'|$. Write the diagonalization of H as $H = \Phi\Lambda\Phi^{-1}$, where $\Phi = [\phi_1, \ldots, \phi_n]$ and $\Lambda = \mathrm{diag}(\mu_1, \ldots, \mu_n)$. Also write $x^{(0)} = \Phi a$, where $a = [a_1, \ldots, a_n]^T$. Then, expanding (2.3) k times, we have

$$\begin{aligned} x^{(k)} &= \alpha^{(0)}\ldots\alpha^{(k-1)}H^k x^{(0)} + \\ &\quad + (\beta^{(0)}\alpha^{(1)}\ldots\alpha^{(k-1)}H^{k-1} + \beta^{(1)}\alpha^{(2)}\ldots\alpha^{(k-1)}H^{k-2} + \cdots + \beta^{(k-1)}H^0)\mathbf{1}. \end{aligned} \tag{2.6}$$

Because $\mathbf{1}$ is an eigenvector of H corresponding to eigenvalue 1, the term in the second line is equal to $\gamma^{(k)}\mathbf{1}$ with $\gamma^{(k)} = \beta^{(0)}\alpha^{(1)}\ldots\alpha^{(k-1)} + \beta^{(1)}\alpha^{(2)}\ldots\alpha^{(k-1)} + \cdots + \beta^{(k-1)}$. For the first term on the right of the first line,

$$H^k x^{(0)} = \Phi\Lambda^k a = a_1\phi_1 + a_2\mu_2^k\phi_2 + \cdots + a_n\mu_n^k\phi_n,$$

with $\phi_1 = \mathbf{1}$. Therefore,

$$x^{(k)} = \alpha^{(0)} \dots \alpha^{(k-1)}(a_1\mathbf{1} + a_2\mu_2^k\phi_2 + \cdots + a_n\mu_n^k\phi_n) + \gamma^{(k)}\mathbf{1},$$

and hence

$$\frac{(x^{(k)})_i - (x^{(k)})_j}{\alpha^{(0)} \dots \alpha^{(k-1)}\mu_2^k} = (e_i - e_j)^T[a_2\phi_2 + a_3(\mu_3/\mu_2)^k\phi_3 + \cdots + a_n(\mu_n/\mu_2)^k\phi_n].$$

(i) When k is large, the term inside the square bracket is dominated by

$$a_2\phi_2 + \cdots + a_t\phi_t,$$

because all other μ_i's are smaller than μ_2 in magnitude. Thus, when $k \to \infty$, only this term remains, hence the result.

(ii) Similar to the above case,

$$\frac{(x^{(2k+p)})_i - (x^{(2k+p)})_j}{\alpha^{(0)} \dots \alpha^{(2k+p-1)}\mu_2^{2k+p}} = (e_i - e_j)^T[a_2\phi_2 + a_3(\mu_3/\mu_2)^{2k+p}\phi_3 + \cdots \\ \cdots + a_n(\mu_n/\mu_2)^{2k+p}\phi_n].$$

When $k \to \infty$, in the square bracket only the term

$$a_2\phi_2 + a_3(\mu_3/\mu_2)^{2k+p}\phi_3 + \cdots + a_t(\mu_t/\mu_2)^{2k+p}\phi_t$$

remains and the rest vanishes. Because the eigenvalues μ_2, ..., μ_t are all real, the ratios μ_3/μ_2, $\dots \mu_t/\mu_2$ can only be ± 1. Hence, taking square, the ratios all become 1. We thus obtain the result in the theorem.

□

Informally speaking, the above theorem states that in the limit, the difference between two elements of the iterate vector $x^{(k)}$ is proportional to that between the corresponding elements of some vector ξ. When $|\mu_2|$ is strictly greater than $|\mu_3|$, this vector ξ may be taken to be the eigenvector ϕ_2. When there exist more than one eigenvalue equal to μ_2, say, $\mu_2 = \cdots = \mu_t$ for some $t > 2$, then

ξ is a linear combination of $\phi_2, \ldots, \phi_t$, where the coefficients of the combination are the expansion coefficients of the initial iterate $x^{(0)}$ along the eigen-basis of the iteration matrix H. However, when there exist more than one eigenvalue whose magnitude is equal to $|\mu_2|$ but these eigenvalues are not all the same, then the situation is slightly complicated. Every other iterate $x^{(k)}$ form a subsequence and the limiting behaviors of these two interleaving subsequences correspond to two different vectors, which are η_0 and η_1 defined in the theorem. Both of them are a linear combination of the eigenvectors $\phi_2, \ldots, \phi_t$. For η_0, the coefficients of the combination are the expansion coefficients of $x^{(0)}$ along the eigen-basis of H; for η_1, some of these coefficients flip signs.

The existence of these various cases is owing to the different choices of ω. Recall that the eigenvalues of $D^{-1}W$ are symmetrically distributed around zero. Then, the eigenvalues of $H = \omega D^{-1}W + (1-\omega)I$ are symmetrically distributed around $1-\omega$, with the smallest one being $1-2\omega$. We will use $\sigma_2 > 0$ to denote the second largest eigenvalue of $D^{-1}W$, which is consistent with the notation in Theorem 2.2.1. By the connected-graph assumption, $\sigma_2 < \sigma_1 = 1$. We also have that the multiplicity of $1-2\omega$ is one, by the same assumption. Furthermore, the second largest eigenvalue of H is $\omega\sigma_2 + 1 - \omega$.

(a) If $\omega > \frac{2}{3-\sigma_2}$, then $1-2\omega$ is negative and it has a larger magnitude than does $\omega\sigma_2 + 1 - \omega$. In such a case, $\mu_2 = 1-2\omega$ and $|\mu_2|$ is strictly greater than $|\mu_3|$. This scenario corresponds to the case (i) of Theorem 2.2.4 with $t = 2$.

(b) If $\omega \leq \frac{1}{2}$, then $1-2\omega$ is nonnegative, and hence all eigenvalues of H are nonnegative. In such a case, $\mu_2 = \omega\sigma_2 + 1 - \omega$. This scenario also corresponds to the case (i) of Theorem 2.2.4, because if there are more than one eigenvalue whose magnitude is equal to μ_2, then the nonnegativity implies that these eigenvalues must be the same as μ_2. The multiplicity of μ_2 is equal to the multiplicity of σ_2. It could happen that either the multiplicity of σ_2 is one, in which case $t = 2$; or the multiplicity of σ_2 is greater than one, in which case $t > 2$.

(c) If $\omega = \frac{2}{3-\sigma_2}$, then $1-2\omega$ is negative, $\omega\sigma_2 + 1 - \omega$ is positive, and the two have the same magnitude. This scenario corresponds to the case (ii) of Theorem 2.2.4. Whether $t = 3$ or $t > 3$ depends on the multiplicity of $\omega\sigma_2 + 1 - \omega$ (equivalently that of σ_2), because the multiplicity of $1-2\omega$ is one. If the multiplicity of σ_2 is one, then only two eigenvalues have magnitude equal to $|\mu_2|$, and hence $t = 3$; otherwise, $t > 3$.

(d) If $\frac{1}{2} < \omega < \frac{2}{3-\sigma_2}$, then $1-2\omega$ is negative and $\omega\sigma_2 + 1 - \omega$ is greater than the magnitude of

$1-2\omega$. This scenario is similar to that of the above item (b), except that not all the eigenvalues of H are nonnegative. All other properties are otherwise the same.

2.2.3 Mutually influenced model

In the standard graph case, the edge weights provide a first-order measurement of vertex similarity/distance. This measure is applicable to only adjacent vertices. Hence, for a pair of vertices that are not adjacent, global information of the graph must be incorporated for extending the measurement. The *algebraic distance on graphs* [52] defines algebraic coordinates for every vertex through an iterative process similar to that in the present work. In the limit, the coordinate difference, which serves the notion of "distance," is proportional to the difference of the corresponding elements of some vector y. Specifically, if we let w_{ij} be the weights of a pair of vertices ij in the graph, then the elements of the vector y satisfy an equilibrium state

$$y_i = \gamma y_i + \sum_j \frac{w_{ij}}{d_i} y_j, \tag{2.7}$$

where $d_i = \sum_j w_{ij}$ is used for normalization and $0 < \gamma < 1$ is a constant factor for all i. The algebraic coordinates do not need to coincide with the y_i's; it suffices for their differences to be proportional. When one treats the vertices to be entities of a mutually influenced environment, then the value y_i of each entity is composed of two components according to (2.7): a portion of itself (γy_i) and a normalized weighted contributions of its neighbors ($\sum_j \frac{w_{ij}}{d_i} y_j$). If y_i denotes the amount of information stored at vertex i, Equation 2.7 essentially signifies a global equilibrium of the flow of information. Such a view is used to interpret the global similarity of vertices through neighborhoods. In short, two vertices are similar if their neighborhoods are similar, because the common factor γ is constant and similarity relies on the neighboring weights w_{ij}.

The present work extends this notion to hypergraphs. In such a setting, the hyperedges cover not just a pair ij of vertices, but rather, a subset of any size (excluding of course empty sets and singletons). Hence, a proxy of the pairwise environment is the star expansion graph G, whose vertex set V' includes not only the original vertices V of the hypergraph, but also the hyperedges E. Naturally, the edges of G come from the containment relation between V and E; that is, $v \in V$ and $h \in E$ are connected if and only if $v \in h$. Therefore, the edge weights of G come from the weights in the original hypergraph, properly scaled. It is important to note that the scaling is

not symmetric, because potentially one hyperedge may contain a large number of vertices. Thus, whereas the weights for the directed edge from v to h are $w(v)$ without scaling, those for the edges from h to v are $w(h)$ scaled by $|h|$.

Then, the *algebraic distance on hypergraphs* enjoys the same equilibrium model Eq. 2.7. From Theorem 2.2.4 and the subsequent discussions, we know that $\gamma = (1 - \mu_2)/\omega$, where μ_2 is the second largest eigenvalue in magnitude of the iteration matrix H, and y is a vector in the subspace spanned by the eigenvectors associated with eigenvalues whose magnitude are equal to $|\mu_2|$. In practice, ω is often set to be $1/2$, then μ_2 is positive[1] and all other eigenvalues having the same magnitude must be equal to μ_2. If furthermore the multiplicity of μ_2 is 1, then the subspace is spanned by the eigenvector ϕ_2 only and hence y can be taken to be ϕ_2. Note that the multiplicity of eigenvalue μ_2 is the same as the multiplicity of the singular value σ_2 of the matrix $(D^h)^{-1/2}(S^h)^{1/2}A(D^v)^{-1/2}(S^v)^{1/2}$ in Theorem 2.2.1.

Finally, when extending the notion of the *algebraic distance* to hypergraphs, we have to take into account the non-pairwise nature of the relations between the vertices. While in pairwise graph setting it is natural to use the simple difference between two vertices' algebraic coordinates as the measure of similarity between them, in non-pairwise hypergraph setting this approach has to be extended. In the hypergraph case we instead assume that the hyperedge is only as important as two most dissimilar vertices in it. In hypergraph partitioning terms, this means that we want to avoid cutting the hyperedges that contain only similar vertices. Therefore we define the algebraic weight of the hyperedge as:

$$alg\_weight(h) = \frac{1}{\max\{|x[v] - x[u]| \mid u, v \in h\}}$$

This way, we penalize cutting the hyperedges that contain only similar vertices, with the notion of similarity defined according to the mutually influenced model.

2.2.4 Experimental Results

In this section, we first illustrate the empirical convergence of Algorithm 1 and then compare the quality of the partitioning produced by our algorithm with those by other state-of-the-art hypergraph partitioners. Our major goal is to study the effectiveness of the algebraic distance on

[1] We assume the nondegenerate case where μ_2 is nonzero. Otherwise, all eigenvalues except for the largest one is zero.

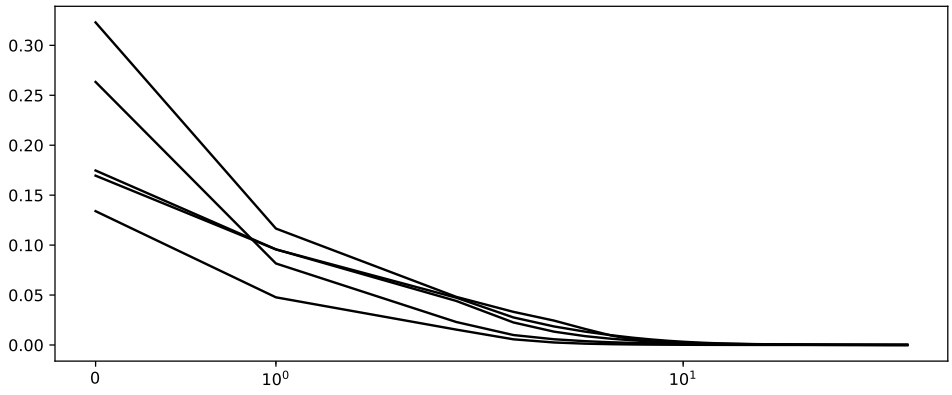


Figure 2.3: Squared sine of the angle between $x^{(k)}$ and $x^{(k+1)}$ as a function of the iteration number k.

hypergraph partitioning by introducing it into a coarsening scheme. We note that these results can clearly be further improved by using more advanced refinement techniques which are beyond the scope of this work.

2.2.4.1 Convergence

The speed of the convergence of algebraic weights depends on the gap between the second largest eigenvalue in magnitude, $|\mu_2|$, of the iteration matrix H and the next eigenvalue with a different magnitude, denoted as $|\mu_{t+1}|$ in the preceding section. Estimating this gap is no less expensive than computing the corresponding eigenvectors. Hence, in practice, we use the squared sine of the angle between two iterates $x^{(k)}$ and $x^{(k+1)}$:

$$1 - \left\langle \frac{x^{(k)}}{||x^{(k)}||}, \frac{x^{(k+1)}}{||x^{(k+1)}||} \right\rangle^2,$$

to measure how parallel the two iterates are, as a proxy of convergence test.

The two consecutive iterates generally become parallel very quickly. In Figure 2.3 we pick five hypergraphs and plot the squared sine for the first 50 iterations. These hypergraphs represent different sizes (from $2426 - by - 3602$ to $103631 - by - 395979$) and different origins (from social networks to circuit simulation) of hypergraphs in the benchmark. One sees that the value is indistinguishable from zero after 10 to 20 iterations. Such a phenomenon is typical to our experience and we generally set *num_iter* comparable to these numbers.

It is worth noting that the parallelism of two consecutive iterates does not necessarily mean true convergence. As noted by Chen and Safro [52], the eigenvalue gap may be so small that it might take a huge number of iterations for the difference

$$(x^{(k)})_i - (x^{(k)})_j$$

(after scaling) to get close enough to $\xi_i - \xi_j$. The parallelism before convergence corresponds to a transient state wherein the change of the iterates is small. It turns out that in practice, a transient state is sufficient for the algebraic distance to be useful for coarsening.

2.2.4.2 Cut improvements

We implement our algorithm with algebraic distances by using the Zoltan [67] package of the Trilinos Project [111]. Zoltan is an open-source toolkit of parallel combinatorial scientific computing algorithms [67] designed to optimize load balancing. It includes a hypergraph partitioning algorithm called PHG (Parallel HyperGraph partitioner). We augment Zoltan's PHG partitioner with algebraic distances as described in the preceding section. Our algorithm is called Zoltan-AlgD. All comparisons with Zoltan are indeed comparisons with Zoltan's PHG partitioner. In this study we use Zoltan in the serial mode. In addition to comparing Zoltan-AlgD to Zoltan, we compare it with two other state-of-the-art partitioners, namely hMetis2 [130] and PaToH v3.2 [46].

PaToH is used as a plug-in for Zoltan, as described in Zoltan's User Guide [68]. We run PaToH with two different parameters: default (here denoted as PaToH-D) and quality (PaToH-Q). hMetis2 is used to directly optimize k-way partitioning. All parameters are set as default: greedy first-choice scheme for coarsening, random k-way refinement, and min-cut objective function.

Each algorithm is run 10 times and the smallest cuts over all runs are compared. In the experiments, standard deviation of the cuts is usually small ($< 5\%$). Interestingly, in a small number of hypergraphs, we observe that the distribution of cuts is bimodal (i.e., the partitioner produces cuts close to either one of two modes). In this case, the standard deviation is high; however, within each mode, the deviation is low. Such behavior demonstrates that in certain settings the solvers cannot escape false local attraction basins obtained at the coarse levels. This hints that the current state of hypergraph partitioning solvers is still far from being optimal and there is a lot of space for improvement.

Because the implementation of our algorithm has not been optimized, the run time of Zoltan-AlgD is on average two to four times longer than that of Zoltan (see Figure 2.8). However, since the iteration scheme is easily parallelizable (for example, among different random vectors, the iteration is pleasingly parallel as well as parallelization of Jacobi-based relaxations has been studied and used in many works), the overhead of computing the algebraic distances may be minimized and thus the run time may be made similar to that of Zoltan. Moreover, the algebraic distance is, in fact, several iterations of Jacobi over-relaxation whose parallelization has been studied a lot.

We also have to point out the lack of progress in coarsening techniques in the recent years, with all major hypergraph partitioning packages using the same approach, making our contribution more valuable. While the way we select vertices to be merged together during coarsening is indeed more expensive than traditional approaches, it introduces very low overhead (*constant* number of passes over vertices) and shows great improvement in the area where state-of-the-art has not changed for a long time.

The different algorithms are compared on three groups of hypergraphs: big-bench, SNAP-bench, and social-networks-bench.

The first two groups (big-bench and SNAP-bench) are generated from matrices obtained from the University of Florida Sparse Matrix Collection [64] by using a row-net model: a vertex i belongs to hyperedge j if there is a non-zero element on the intersection of ith column and jth row. Big-bench contains 443 matrices; many of them are derived from optimization problems. SNAP-bench consists of 42 matrices from the SNAP (Stanford Network Analysis Platform) Network Data Sets [140].

The third group of the benchmark, social-networks-bench (12 hypergraphs), has two parts. The first part contains two networks with known communities (youtube and flickr) from the IMC 2007 Data Set [160]. The hypergraphs are constructed in the following way: each vertex represents a user and each hyperedge represents a community. After generating the hypergraph, isolated vertices are removed. The second part contains "similar hypergraphs." They are generated by using the following pipeline: a graph of pairwise links is obtained for the same dataset and then a similar graph is generated by using the BarabasiAlbertGenerator [24, 26] of NetworKit [234]. Afterwards, the adjacency matrix of the new graph is interpreted by using the row-net model.

In Figures 2.4,2.5,2.6 and 2.7 the results are presented graphically. Each curve plots the

ratio

$$\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}},$$

where for the light grey "X" markers curve, the "other algorithm" is Zoltan; black triangle markers curve, PaToH-Q; dark grey dot markers curve, PaToH-D; and dim grey "Y" markers curve, hMetis2. Each plot corresponds to a certain number of parts and a certain imbalance factor. The hypergraphs are ordered in the the increasing ratio. Colorblind-safe color versions of the figures are available online at `http://bit.ly/relaxation2019`.

For readability, the results for big-bench are split in two parts: those with ratios

$$\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} < 1.5$$

and those with ratios greater than 1.5 (i.e., with more than 50% improvement in cut). The results for the social-networks-bench and SNAP-bench are plotted on the same figure, since the social-networks-bench is considerably smaller than the other two. The results show substantial improvement over Zoltan without algebraic distance, as well as over hMetis2 and PaToH on most of the hypergraphs. For the full set of results, please refer to `http://shaydul.in/hypergraph-partitioning-archive/`.

2.3 Aggregative Coarsening for Multilevel Hypergraph Partitioning

The standard approach to coarsening used in most state-of-the-art hypergraph partitioners is matching-based (see discussion in Section 2.1.2). Originally, this meant that at each level pairs of adjacent vertices are selected to become one vertex at the next level. This technique has later been extended to include non-pairwise matchings (i.e., more than two fine vertices can form a coarse vertex). One of the alternative approaches is aggregative coarsening inspired by algebraic multigrid. In aggregative coarsening, the set of vertices V is separated into disjoint sets of seeds and non-seeds, namely, C and F such that $F \cup C = V$. The non-seed vertices aggregate themselves around the seeds (hence the name aggregative coarsening). The aggregation can be strict (F-vertices are not split) and weighted (F-vertices can be split between multiple seeds with vertex weight conservation). At the refinement stage, the partitioning decision (i.e., partition assignment) is interpolated from

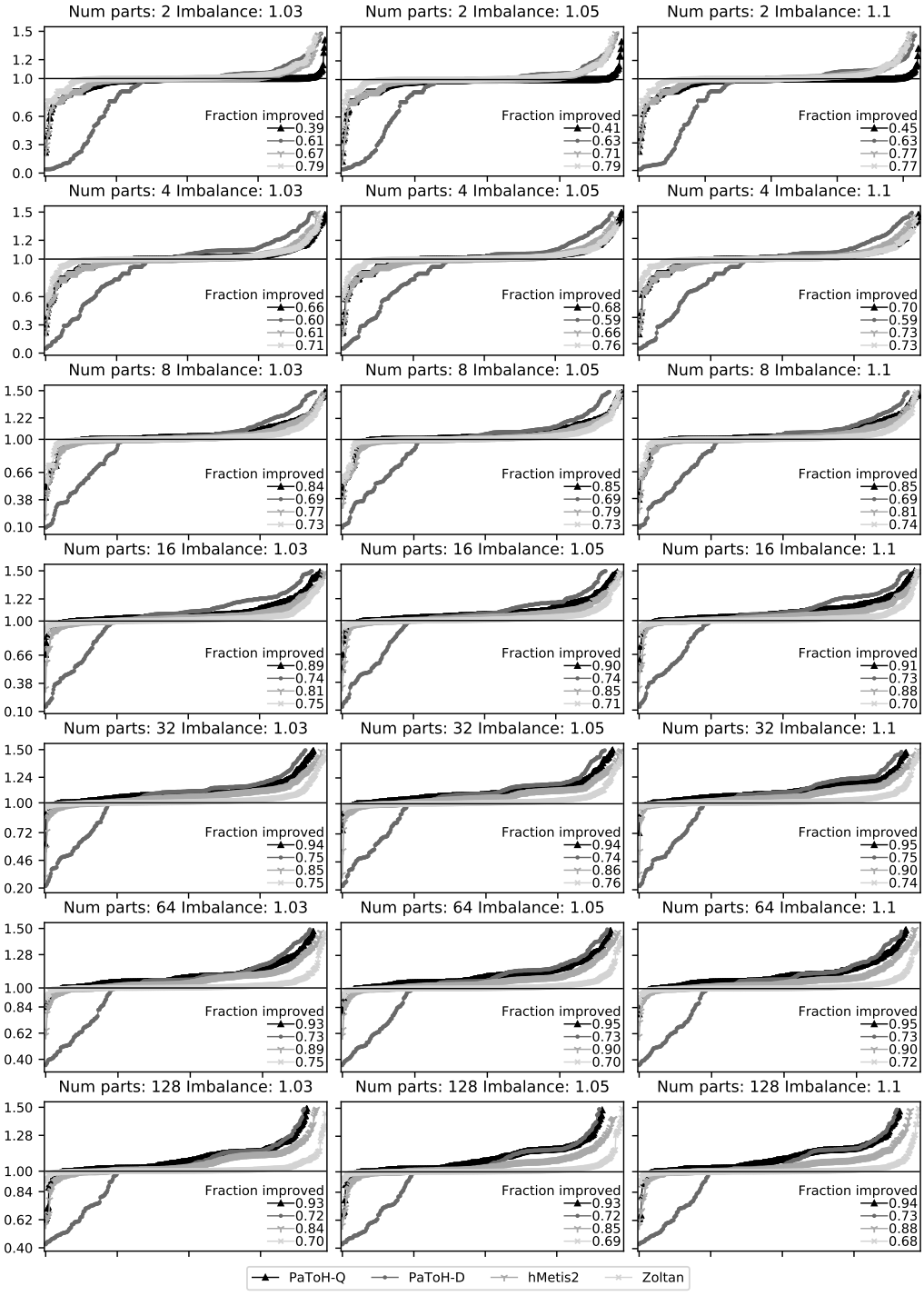


Figure 2.4: First half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} < 1.5$) of big-bench (443 hypergraphs), with light grey "X" markers curve corresponding to Zoltan, black triangle markers to PaToH-Q, dark grey dot markers to PaToH-D, and dim grey "Y" markers to hMetis2. The improvements are with respect to the whole big-bench, including those with improvement greater than 1.5.

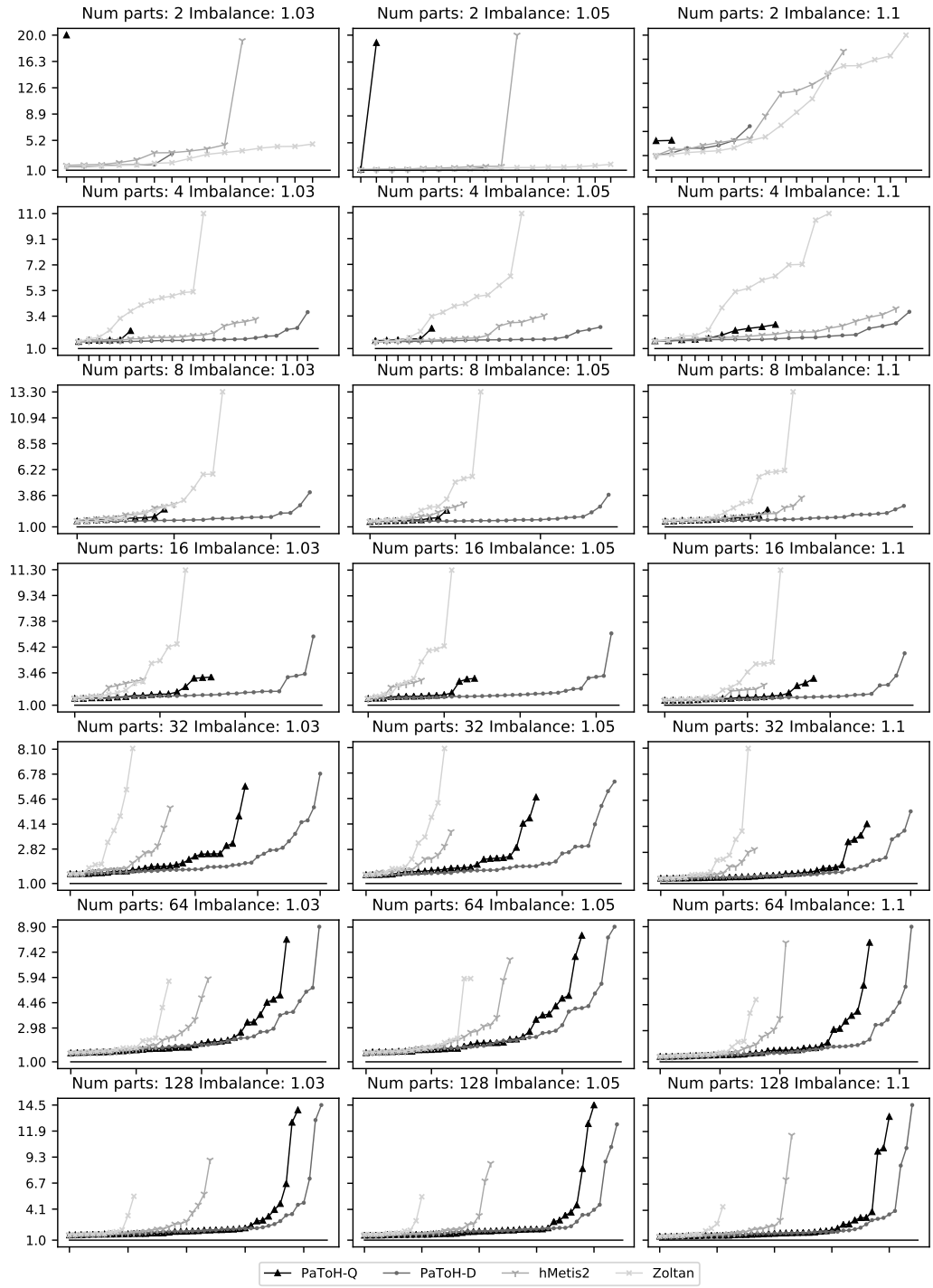


Figure 2.5: Second half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} \geq 1.5$) of big-bench (443 hypergraphs), with light grey "X" markers curve corresponding to Zoltan, black triangle markers to PaToH-Q, dark grey dot markers to PaToH-D, and dim grey "Y" markers to hMetis2 (greater is better).

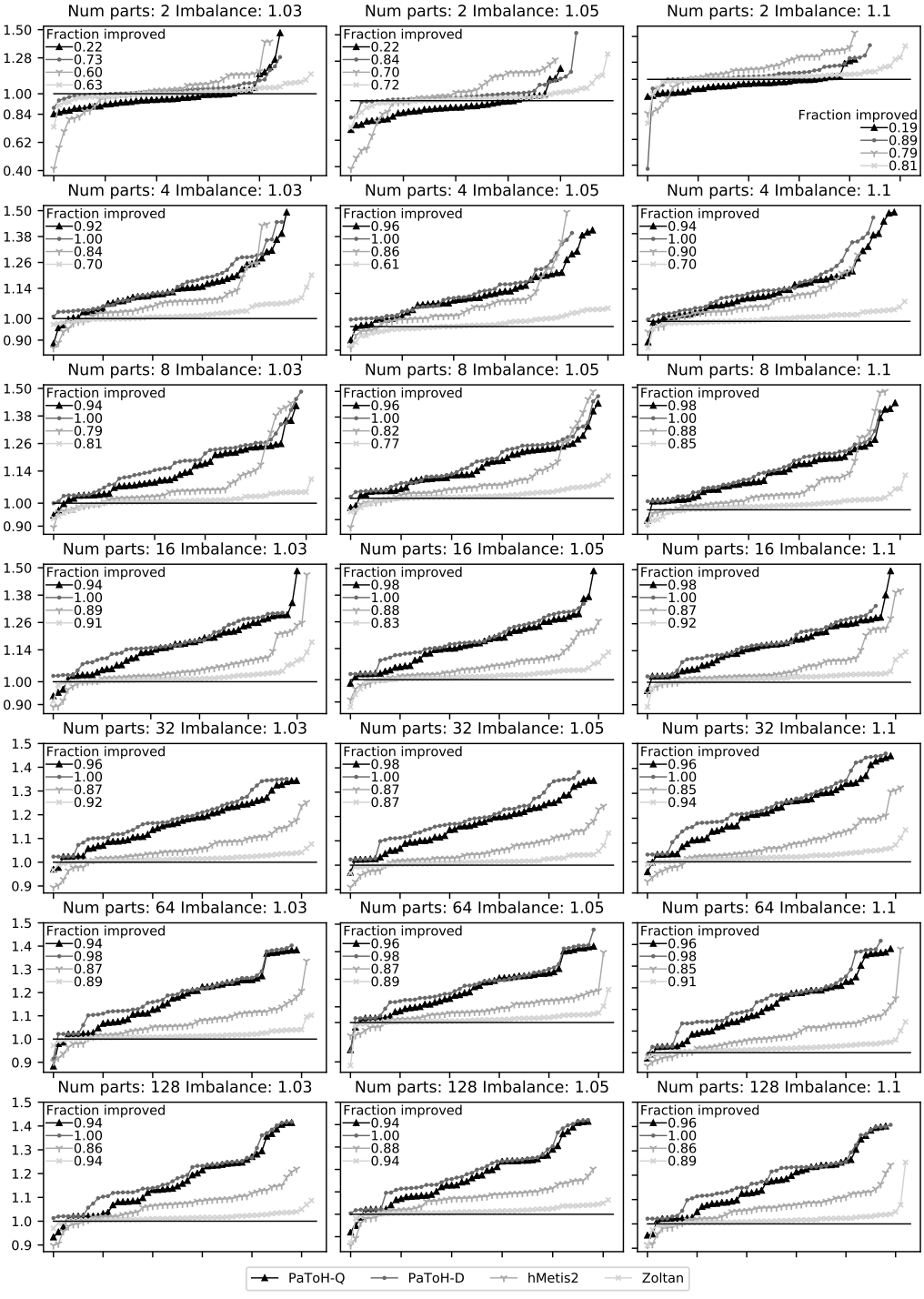


Figure 2.6: First half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} < 1.5$) of SNAP-bench and social-networks-bench (54 hypergraphs), with light grey "X" markers curve corresponding to Zoltan, black triangle markers to PaToH-Q, dark grey dot markers to PaToH-D, and dim grey "Y" markers to hMetis2. The improvements are with respect to the whole big-bench, including those with improvement greater than 1.5

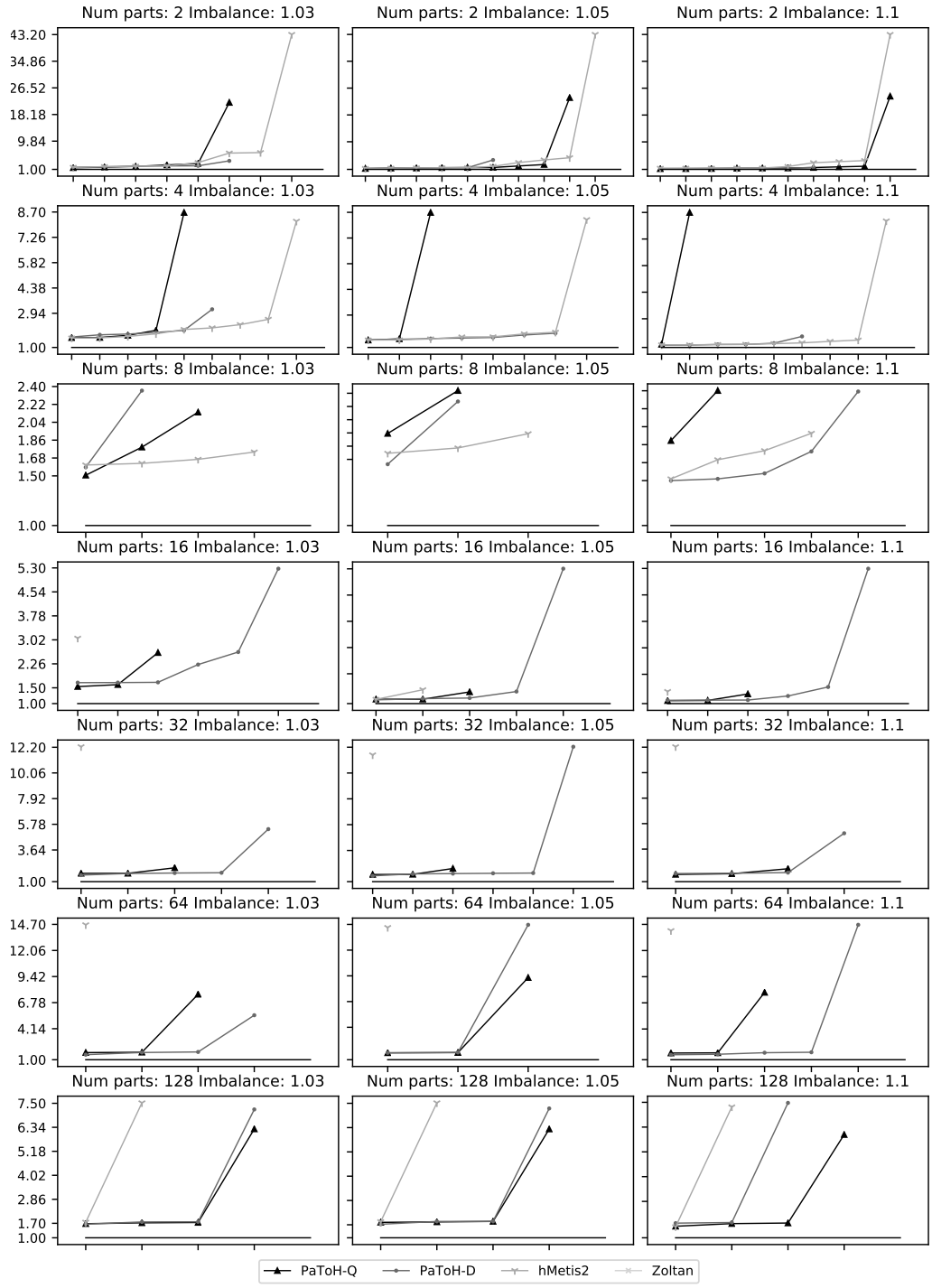


Figure 2.7: Second half ($\frac{\text{cut obtained using another algorithm}}{\text{cut obtained using Zoltan-AlgD}} \geq 1.5$) of SNAP-bench and social-networks-bench (54 hypergraphs), with light grey "X" markers curve corresponding to Zoltan, black triangle markers to PaToH-Q, dark grey dot markers to PaToH-D, and dim grey "Y" markers to hMetis2 (greater is better).

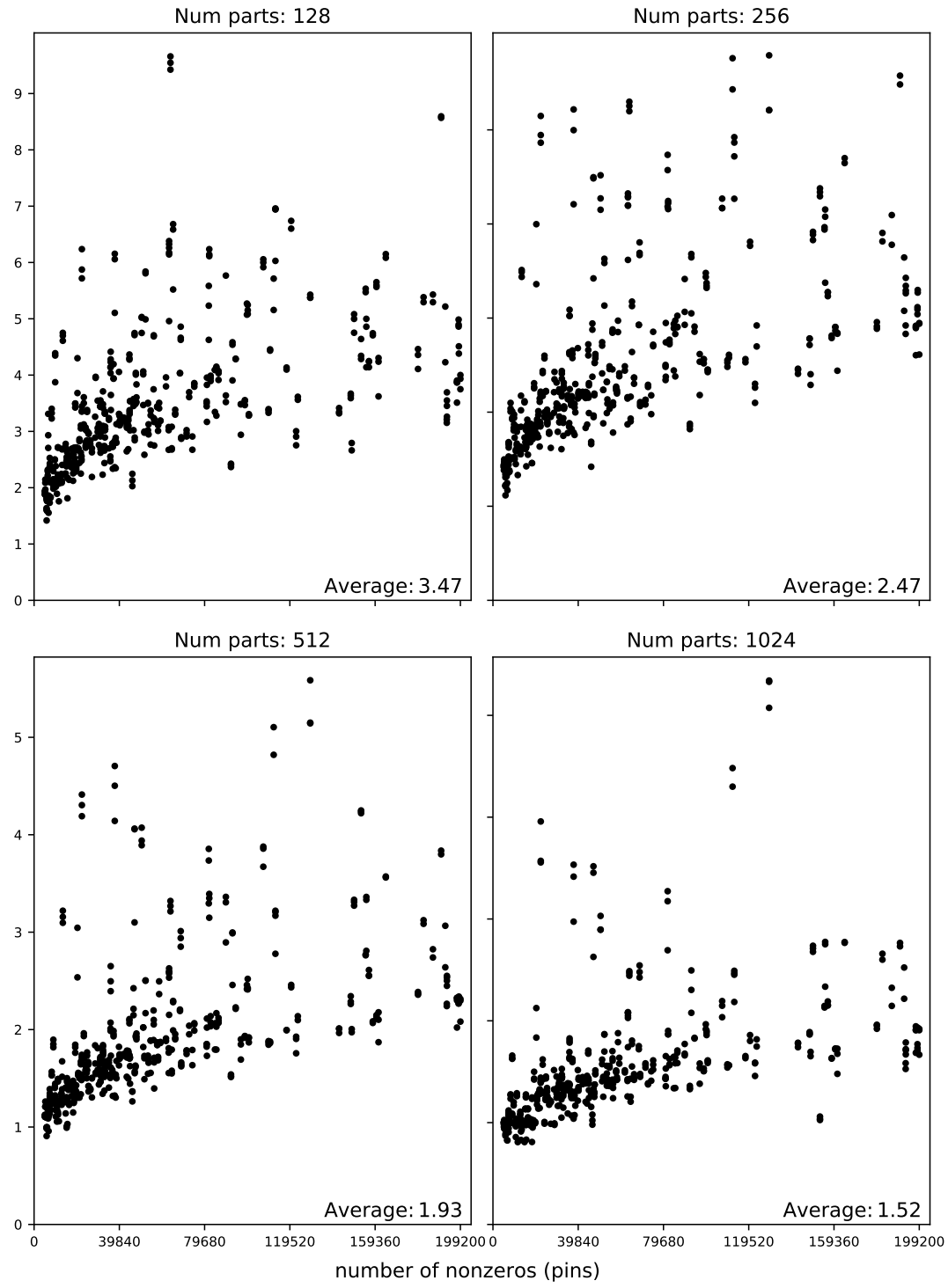


Figure 2.8: Ratio $\frac{\text{Zoltan-AlgD runtime}}{\text{Zoltan runtime}}$ for the big-bench benchmark. Time is CPU time as reported by *zdrive*.

each seed to the non-seeds in its aggregate. This separation between seed and non-seeds helps to introduce additional guarantees. For example, on graphs Safro et al. [213] introduce the notion of strong connection and guarantee that each vertex in the graph is strongly connected to at least one seed. The weighted aggregation was initially introduced for several cut problems on graphs [212, 205, 214] including GP [215]. There was an unfinished attempt to extend this approach to hypergraphs. Buluç and Boman [44] describe several challenges in applying aggregative coarsening to hypergraphs, as well as propose two very similar coarsening schemes, strict and weighted. In this work, we limit our discussion to strict aggregation.

In aggregative coarsening, two main questions have to be addressed: seed selection and aggregation of non-seeds around seeds. In the process of seed selection, Buluç and Boman [44] follow Safro et al. [213] in using the concept of *future volumes*. Future volume is a measure of how many vertices a seed can incorporate into itself (in other words, how large a vertex can grow). They propose computing future volumes on the star expansion of the hypergraph (thus limiting the complexity), then iteratively adding vertices with high future volumes to the set of seeds C until $|C|$ reaches a certain threshold. Aggregation rules are established on the star expansion of the hypergraph. Seeds and non-seeds select a constant number of adjacent hyperedges to "invade" based on the exclusive coarseness (a metric indicating how many seeds an hyperedge contains).

2.3.1 Two Aggregation Algorithms

Our algorithm combines the ideas of aggregative coarsening described in [213] and [44] with the algebraic distance [205, 223] presented in Section 2.2. Aggregative coarsening is a two-step process, so we have to address both the seed selection and the rules of aggregation. At each coarsening level, a set of seeds is selected and each seed is assigned a set of non-seeds to form a cluster. The cluster at a given coarsening level becomes one vertex at the next level.

Both introduced schemes utilize algebraic distances by augmenting hyperedge weights with algebraic weights. We define the algebraic weight of hyperedge e as an inverse of the algebraic distance between two farthest apart vertices in e, i.e.,

$$\rho(e) = 1 \;/\; \max_{i,j \in e} \text{algdist}_{ij}. \tag{2.8}$$

2.3.1.1 Seed selection

For the seed selection we utilize two core concepts: *future volumes* and *strong connection*. The main goal is to construct a set of seeds C such that every vertex in the graph is *strongly connected* to C. We define *strong connection* as follows: the vertex $i \in F$ is *strongly connected* to C if the sum of algebraic weights of the edges connecting it to C is more than a certain fraction of the total algebraic weight of incident edges:

$$i \text{ is strongly connected to } C \qquad \Longleftrightarrow \qquad \frac{\Sigma_{j \in C} \rho(e_{ij})}{\Sigma_j \rho(e_{ij})} > Q, \tag{2.9}$$

where Q is a parameter (in our experiments $Q = 0.5$). The *future volume* of a vertex is a measure of how large an aggregate seeded by it can grow. Intuitively, we want to add the vertices with very high volume (or the ones that might become centers of the aggregates of very high volume) to the set of seeds. *Future volume* of a vertex is defined as follows (note that here we use the hyperedge weights w and not the algebraic weights ρ):

$$fv(i) = w(i) + \Sigma_j w(j) \frac{w(e_{ij})}{\Sigma_k w(e_{jk})}. \tag{2.10}$$

We begin the construction of set C by computing future volumes for all vertices. Then, we initialize C with vertices with large future volumes (if mean future volume is m_{fv} and standard deviation of the distribution of future volumes is σ_{fv}, then $i \in C \iff fv(i) > m_{fv} + 2\sigma_{fv}$) and initialize F with all other vertices, such that $F \cup C = V$. After that the future volumes of vertices in F are recomputed, only taking into account connections with other vertices in F (i.e., in Equation (2.10) assume $w(e_{ij}) = 0$ if $j \in C$ or $i \in C$). Finally, vertices in F are visited in order of decreasing future volume and added to the set C if they are not strongly connected to C. Note that at the end of this process each vertex in V is strongly connected to the set C and $F \cup C = V$. Pseudocode for this procedure is presented in Listing 2.1.

2.3.1.2 Aggregation

We investigate two approaches to establishing the rules of aggregation. First approach is a scheme similar to inner-product matching used in Zoltan[48] and PaToH[47] but applied in algebraic multigrid setting. Second approach consists of computing a stable assignment [97] between vertices

Listing 2.1: Seed selection

```
for i in V:
        fv[i] = w[i] + Σ_j w[j](w[e_ij] / Σ_k w[e_jk])
for i in V:
        if fv[i] > mean(fv) + 2 * stdev(fv):
        C.insert(i)
    else:
        F.remove(i)
for i in F:
        fv[i] = w[i] + Σ_{j∈F} w[j](w[e_ij] / Σ_{k∈F} w[e_jk])

for i in sort_indices(fv):
        if Σ_{j∈C} w[e_ij] / Σ_j w[e_ij] < Q:
                C.insert(i)
                F.remove(i)
```

Listing 2.2: Inner-product aggregation

```
for i in F:
        j = argmax_{u∈C} ipm(v,u)
        C_j.insert(i)
```

of C and F. Both approaches take advantage of algebraic distances as a similarity measure when establishing aggregation rules.

Inner-product aggregation proceeds by visiting the non-seed vertices in the random order. For each unmatched vertex $v \in F$, a neighboring seed $u \in C$ with the highest inner product is selected and v is added to the cluster C_u seeded by it. The inner product is defined as the total algebraic weight of the edges connecting v with the seed u. Concretely, $\text{ipm}(v, u) = \Sigma_{e|v,u\in e}\, \rho(e)$. See Listing 2.2 for pseudocode. We experimented with visiting the non-seeds in order of decreasing future volume and with using connectivity to make decisions when establishing aggregation rules. These approaches are more computationally intensive and do not produce better results (see Section 2.3.3 for the comparison of different parameters).

Stable assignment aggregation begins by constructing preference lists. Each seed orders adjacent non-seeds in the order of decreasing total algebraic weight of the hyperedges connecting them (and vice versa): $\text{pref}_i(j) = \Sigma\rho(e_{ij})$. Then the stable assignment is computed using an algorithm similar to the classical one described in Ref. [82]. Each seed in C proposes to non-seeds in its preference list. If the non-seed does not have a better offer, it tentatively accepts the proposal and is put on the waitlist. If that non-seed later receives a better offer (i.e., an offer from a seed that ranks higher on its preference list), it rejects the current offer and the rejected seed proposes to the next candidate on its preference list. To discourage the creation of very large clusters, we

Listing 2.3: Stable matching aggregation

```
def propose(i):
        for j in pref_list[i]:
                if waitlist[i].size > threshold:
                        return
                if propos[j] == -1: // j holds no proposal
                        propos[j] = i
                        waitlist[i].push_back(j)
                        continue
                if i is preferable to propos[j]:
                        rejected = propos[j]
                        propos[j] = i
                        waitlist[i].push_back(j)
                        propose(rejected)

// Step 1: compute preference lists
for i in F:
        for j in seed_neighbors[i]:
                pref[j] = Σρ(e_ij) // pref is a hashtable
        for j in sort_by_value(pref).keys():
                pref_list[i].push_back(j)
for i in C:
        for j in non_seed_neighbors[i]:
                pref[j] = Σρ(e_ij)
        for j in sort_by_value(pref).keys():
                pref_list[i].push_back(j)

// Step 2: compute stable assignment
for i in C:
        propose(i)
```

limit the size of waitlist for a seed to the maximal vertex weight on a given coarsening level times three plus ten: $\text{len(waitlist)} = 3 \times \text{max\_vtx\_wgt} + 10$. Procedure terminates when each non-seed has been assigned to a waitlist or a seed has been rejected by every non-seed. At this point each seed forms a cluster with all vertices on its waitlist, subject to size constraint (we guarantee that no cluster can be larger than total vertex weight over the number of parts). The fact that we use a classical problem as a subproblem in our heuristic allows us to potentially leverage the previous work in optimizing and parallelizing stable assignment, such as [147],[151] and [87]. The pseudocode is presented in Listing 2.3.

2.3.2 Results

We implemented all algorithms described in this section within Zoltan [67] package of the Trilinos Project [111]. Zoltan is an open-source toolkit of parallel combinatorial scientific comput-

ing algorithms [67]. It includes a hypergraph partitioning algorithm PHG (Parallel HyperGraph partitioner) and interfaces to PaToH and hMetis2. We added our new coarsening schemes and left other phases of the multilevel framework unchanged. Our implementation, data, and full results are available at `http://bit.ly/aggregative2018code`.

The hypergraphs in our benchmark are generated from a selection of matrices using the row-net model. In the row-net model, each column of the matrix represents a vertex, each row represents an edge and a vertex j belongs to the hyperedge i if there is a non-zero element at the intersection of j-th column and i-th row, i.e., $A_{ij} \neq 0$. All matrices (more than 300) were obtained from SuiteSparse Matrix Collection [64] that includes other collections. For each combination of hypergraph/algorithm/set of parameters, we executed 20 experiments.

We compare our algorithm with four state-of-the-art partitioners: hMetis2 [130], PaToH v3.2 [46], Zoltan PHG [67] and Zoltan-AlgD [223]. PaToH is used as a plug-in for Zoltan with default parameters described in Zoltan's User Guide [68]. hMetis2 is used in k-way mode with all parameters set to default: greedy first-choice scheme for coarsening, random k-way refinement, and min-cut objective function. The reason we run hMetis in k-way mode is the way hMetis specifies imbalance constraint. In recursive bisection mode, the imbalance constraint is applied *at each bisection step*, therefore relaxing the constraint as the number of parts increases. We found it almost impossible to compare hMetis in recursive bisection mode fairly (i.e., with the same imbalance) with other partitioners. Both Zoltan and PaToH are used in serial mode.

Optimizing the constants in the running time of the proposed algorithms is beyond the scope of this work. Currently, for the existing unoptimized implementation the running time of other state-of-the-art hypergraph partitioners is not improved except for those that generate less levels in the hierarchy. In the experiments, the runtime of unoptimized implementation of our algorithms is up to an order of magnitude larger than the runtime of other state-of-the-art partitioners in worst cases. However, we must point out that our algorithm utilizes the building blocks and ideas of algebraic multigrid, which makes it possible to improve the runtime drastically by leveraging a plethora of existing research in optimizing and parallelizing algebraic multigrid solvers(e.g. [263], [188]). Similarly, there exists extensive research into optimizing the performance of stable matching solvers. Manne et al. [151] demonstrate the connection between graph matchings and stable marriage and show the scalability of Gale-Shapely type algorithms. Munera et al. [165] present an adaptive search formulation of stable marriage problem and take advantage of a Cooperative Parallel Local

Search framework [164], achieving superlinear speedup. Gelain et al. [85] demonstrate a different efficient local search method for stable marriage problem.

Figures 2.11, 2.12, 2.13, 2.14, 2.9 and 2.10 present experimental results comparing the proposed schemes with state-of-the-art hypergraph partitioners. Figures 2.11 and 2.12 present the results for imbalance factor 3%, Figures 2.13 and 2.14 for imbalance factor 5% and Figures 2.9 and 2.10 for imbalance factor 10%. In Figures 2.11, 2.13 and 2.9, we show the results of inner-product algebraic multirgid aggregation coarsening. In Figures2.12, 2.14 and 2.10, the stable matching aggregation is demonstrated. We use frequency histograms to present the distribution of cut differences between our methods and other state-of-the-art hypergraph partitioners. The value being represented (see horizontal axes) is the ratio

$$\zeta = \frac{\text{cut obtained using another partitioner}}{\text{cut obtained using our method}}. \tag{2.11}$$

Each bin corresponds to a range of the ratios (for example, the middle bin corresponds to the differences of less than $\pm 5\%$ and the rightmost to the improvements of $> 20\%$). Each rectangle corresponds to a partitioner: blue corresponds to PaToH, red corresponds to hMetis2, green corresponds to Zoltan PHG and cyan corresponds to Zoltan-AlgD. For the full results, please refer to `http://bit.ly/aggregative2018results`

Figures 2.11 and 2.12 present the results for imbalance factor 3%, Figures 2.13 and 2.14 for imbalance factor 5% and Figures 2.9 and 2.10 for imbalance factor 10%.

Figure 2.15 presents comparison of the quality of the solutions for the two proposed coarsening methods. Analogously to other barcharts, this figure compares the two algorithms by presenting the value

$$\zeta = \frac{\text{cut obtained using stable matching aggregation}}{\text{cut obtained using inner-product aggregation}}. \tag{2.12}$$

The figure demonstrates that two algorithms produce solutions of very similar quality.

The results demonstrate that given the same refinement, the proposed schemes are at least as effective as traditional matching-based schemes, while outperforming them on many instances. Both proposed coarsening schemes almost equally succeed in improving the quality of solvers. This is be due to the fact that the two methods are equivalent if the waitlist for a seed in stable matching scheme in unlimited in size. In this work, we limited the size of the waitlist, making the two

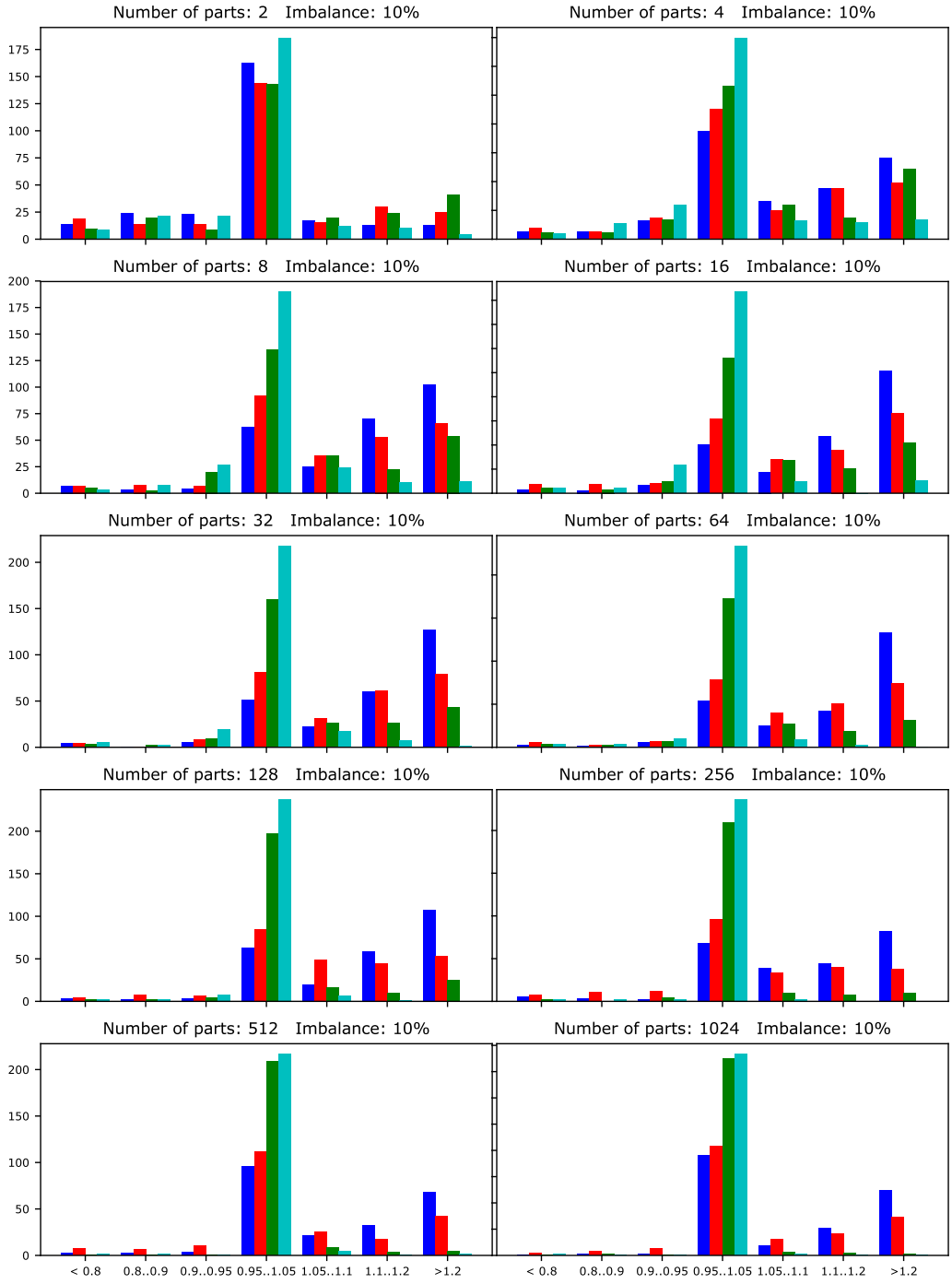


Figure 2.9: Histogram of ζ for coarsening using algebraic multigrid inner-product aggregation with imbalance factor 10%. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

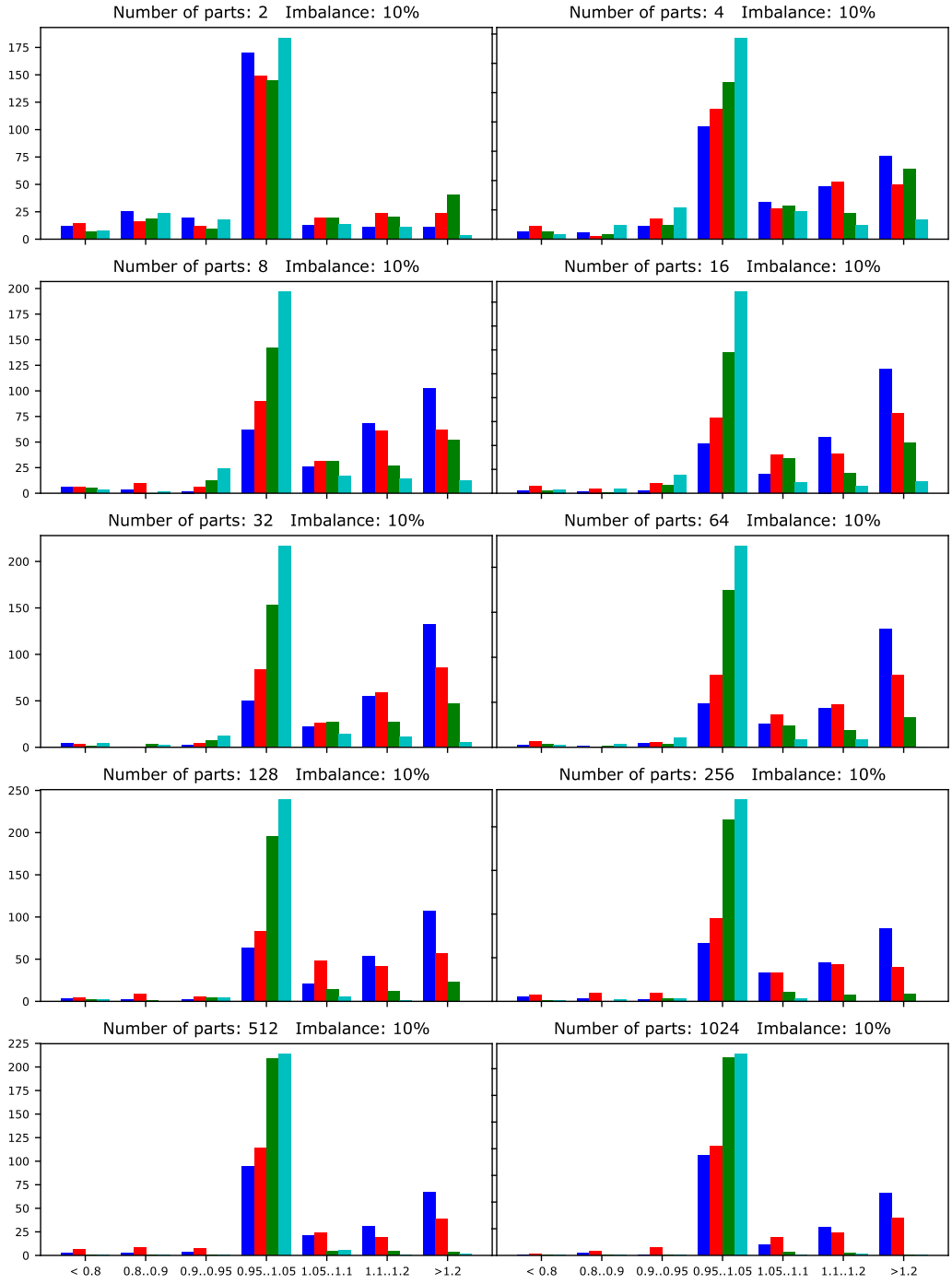


Figure 2.10: Histogram of ζ for coarsening using stable matching aggregation with imbalance factor 10%. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

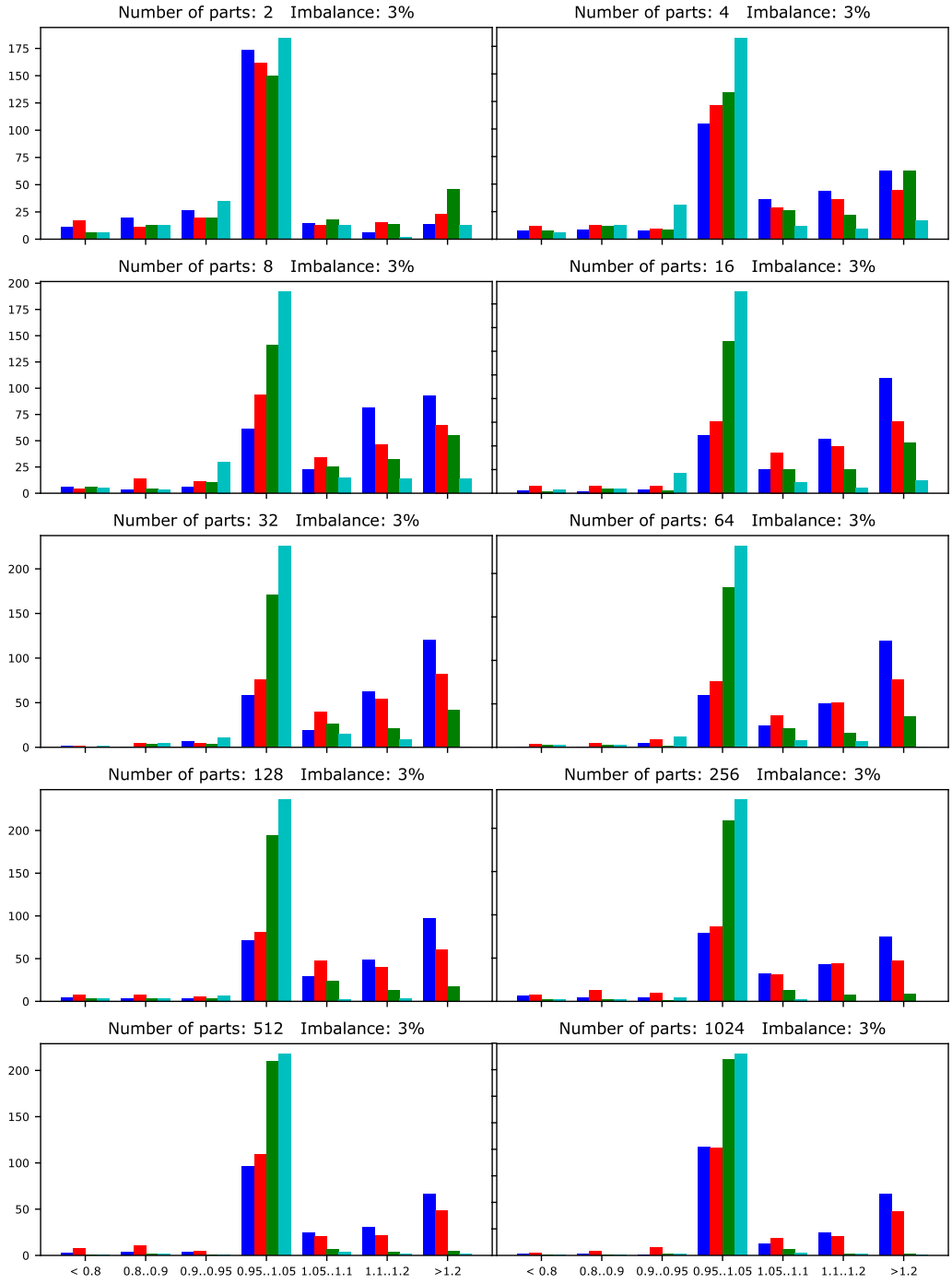


Figure 2.11: Histogram of ζ for coarsening using inner-product aggregation with imbalance factor 3%. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

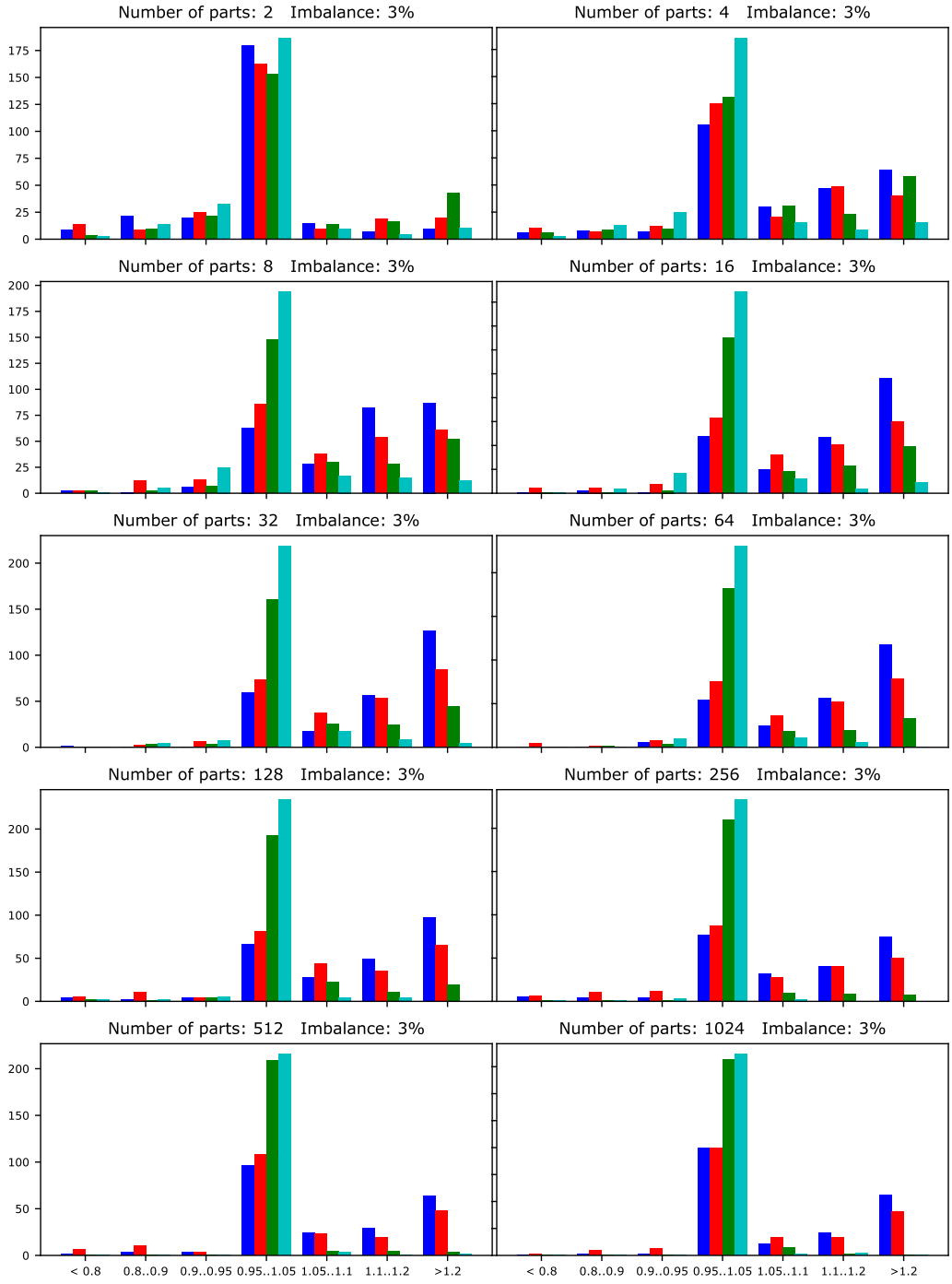


Figure 2.12: Histogram of ζ for coarsening using stable matching aggregation with imbalance factor 3%. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

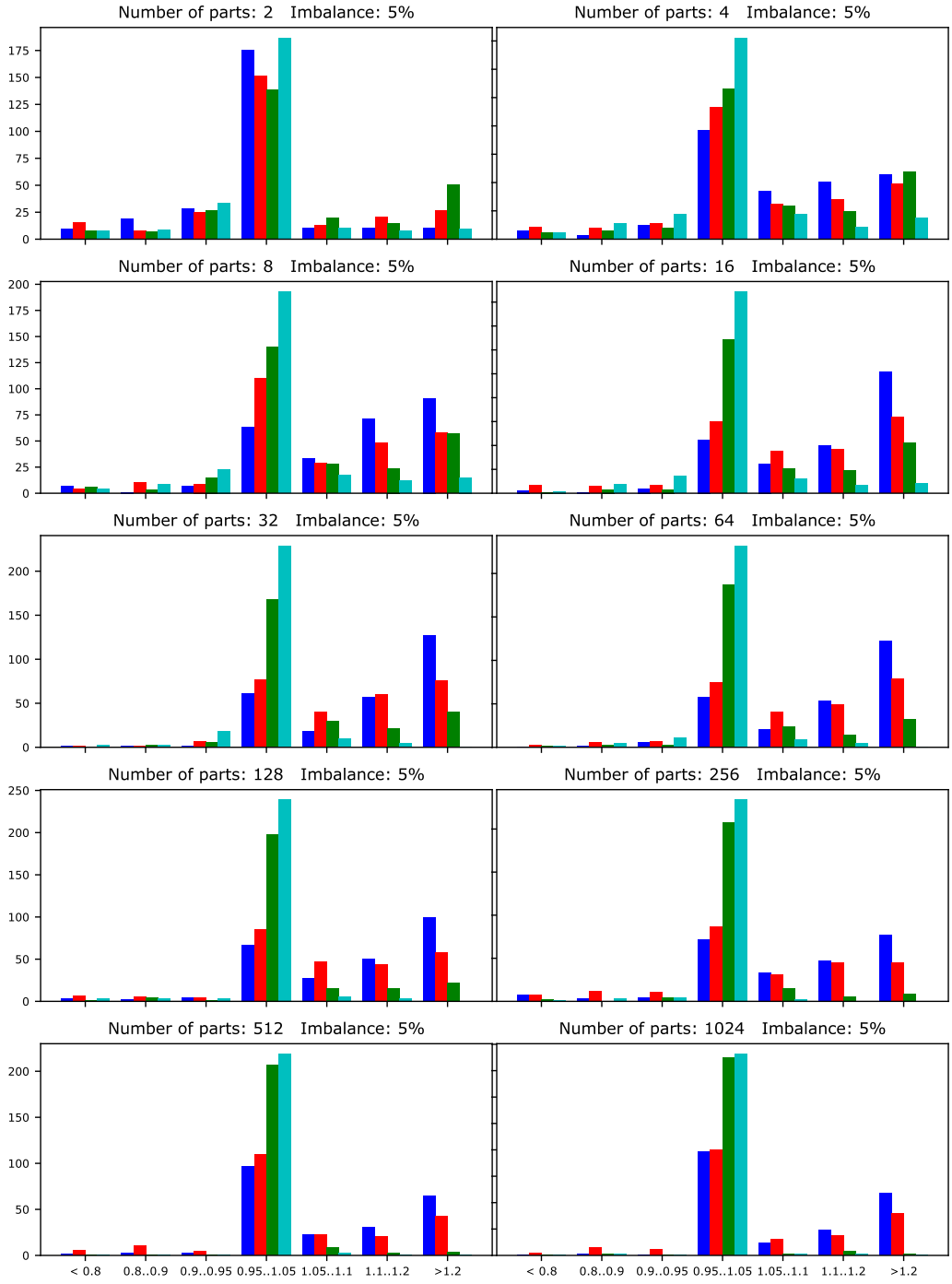


Figure 2.13: Histogram of ζ for coarsening using inner-product aggregation with imbalance factor 5%. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

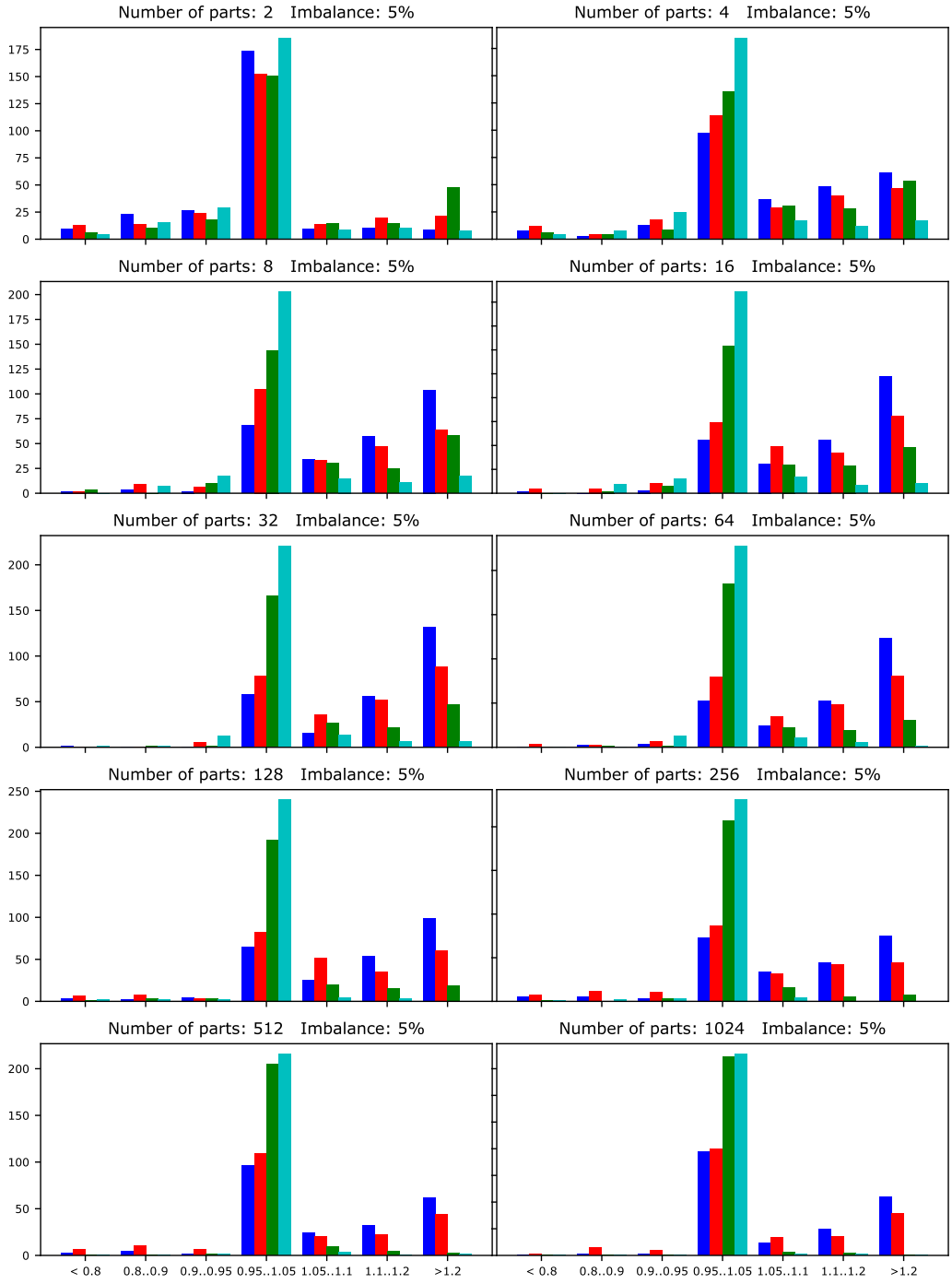


Figure 2.14: Histogram of ζ for coarsening using stable matching aggregation with imbalance factor 5%. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

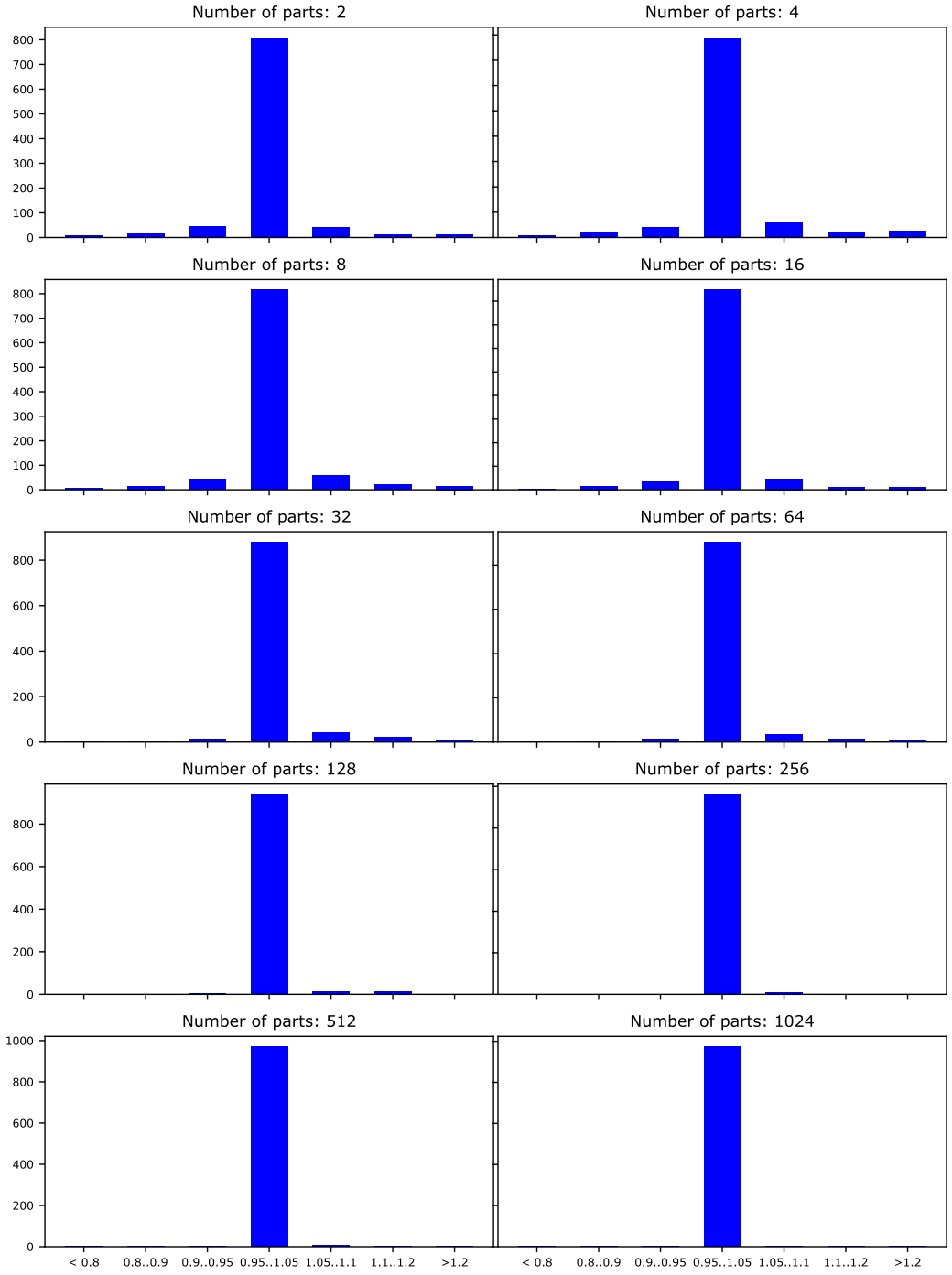


Figure 2.15: Histogram of ζ comparing coarsening using inner-product aggregation (denominator of ζ) with coarsening using stable matching aggregation (nominator of ζ) with imbalance factor 10%. It is easy to see that the two algorithms perform very similarly.

aggregation strategies non-equivalent. However, the limitation on the size of the waitlist is not triggered often (i.e. most waitlists are short), resulting in very similar performance. Since Zoltan utilizes recursive bisectioning scheme, we can see that improvements decrease as number of parts increases. This can be attributed to refinement becoming more and more important as number of parts increases.

2.3.3 Interesting observation about algorithmic variations and additional parameters for algebraic multigrid inner-product aggregation

We explore two additional approaches to inner-product aggregation (see Listing 2.2). First, instead of visiting the vertices in random order, we investigate visiting them in the order of decreasing future volume. Second, instead of using the inner-product as a metric when selecting a seed to join, we explore using the connectivity metric: each non-seed v a neighboring seed u with the highest connectivity is selected and v is added to the cluster C_u seeded by it. The connectivity is defined as $N_{v,C_u}/W_{v,C_u}$, where $N_{v,C_u} = \Sigma_{e|C_u \cap e \neq \emptyset, v \in e}\, \rho(e)$ is the total algebraic weight of the edges connecting v with the vertices in the cluster C_u, and $W_{v,C_u} = \Sigma_{i \in C_u} w(i) + w(v)$ is the total weight of the vertices in the potential cluster [47]. The results of combinations of these two variations are presented in Figures 2.16, 2.17 and 2.18. All include imbalance factors of 10%.

Surprisingly for us, we observed relatively insignificant differences in the results of these two (on the first glance) very important variations. The variation related to the strength of connectivity that depends on the capacity of already chosen cluster directly affects the size of the coarse aggregate. Too big aggregates can result in additional work (and thus computational time) of the refinement and getting trapped in false local attraction basins with KL/FM refinement. However, we observe that the entire framework resolves this issue without any problems.

2.4 Discussion

In this section we have introduced a new similarity measure for hypergraph vertices—algebraic distances, and two novel aggregative coarsening schemes for hypergraphs. This similarity measure is used for more accurate vertex aggregation during the coarsening stage of a multilevel algorithm. A serial iterative procedure for computing algebraic distances is introduced and implemented within the multilevel hypergraph partitioning framework Zoltan. The procedure results in

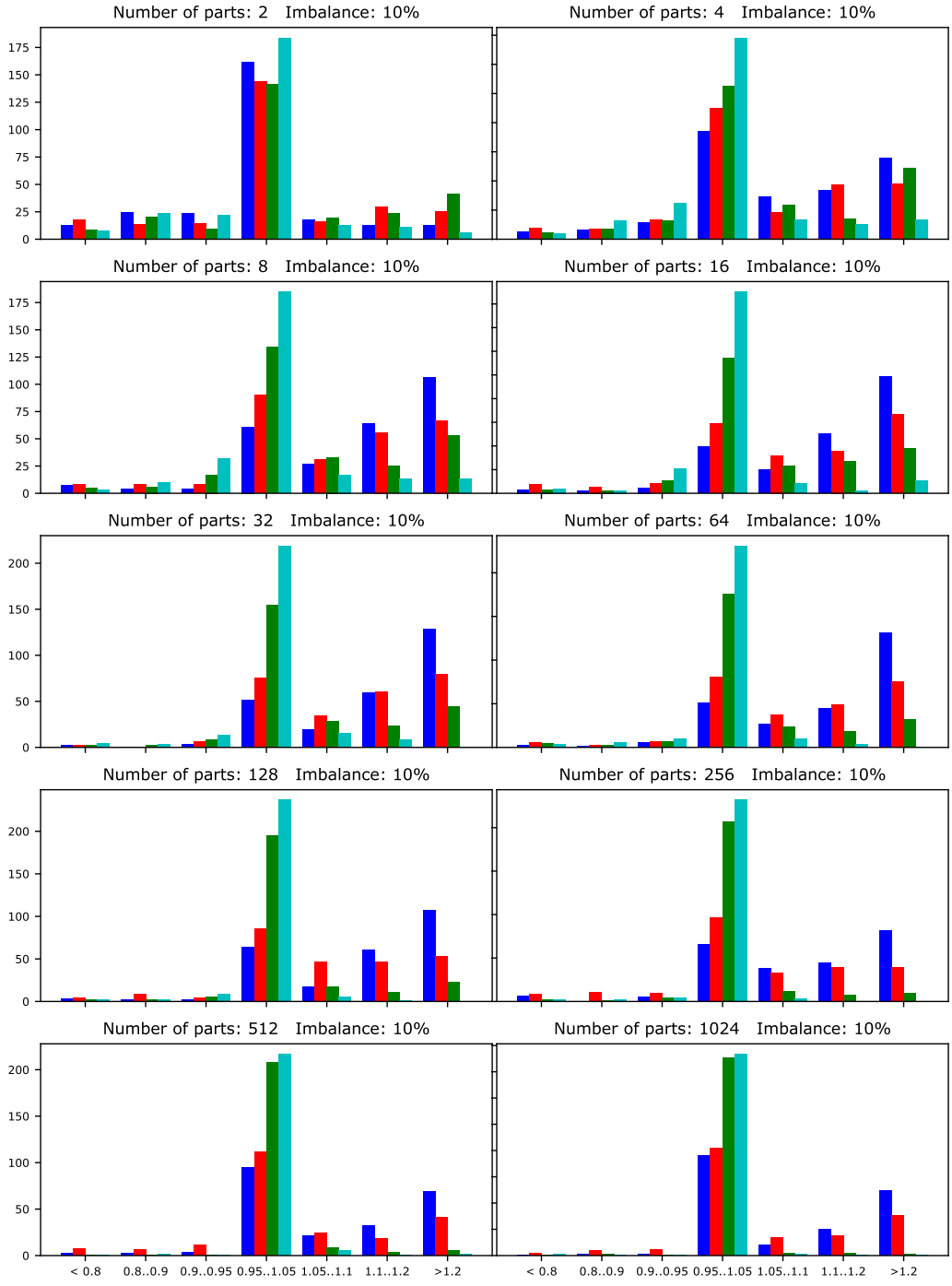


Figure 2.16: Histogram of ζ for coarsening using inner-product aggregation with additional parameters. Vertices are visited in the order of decreasing future volumes, inner-product metric is used. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

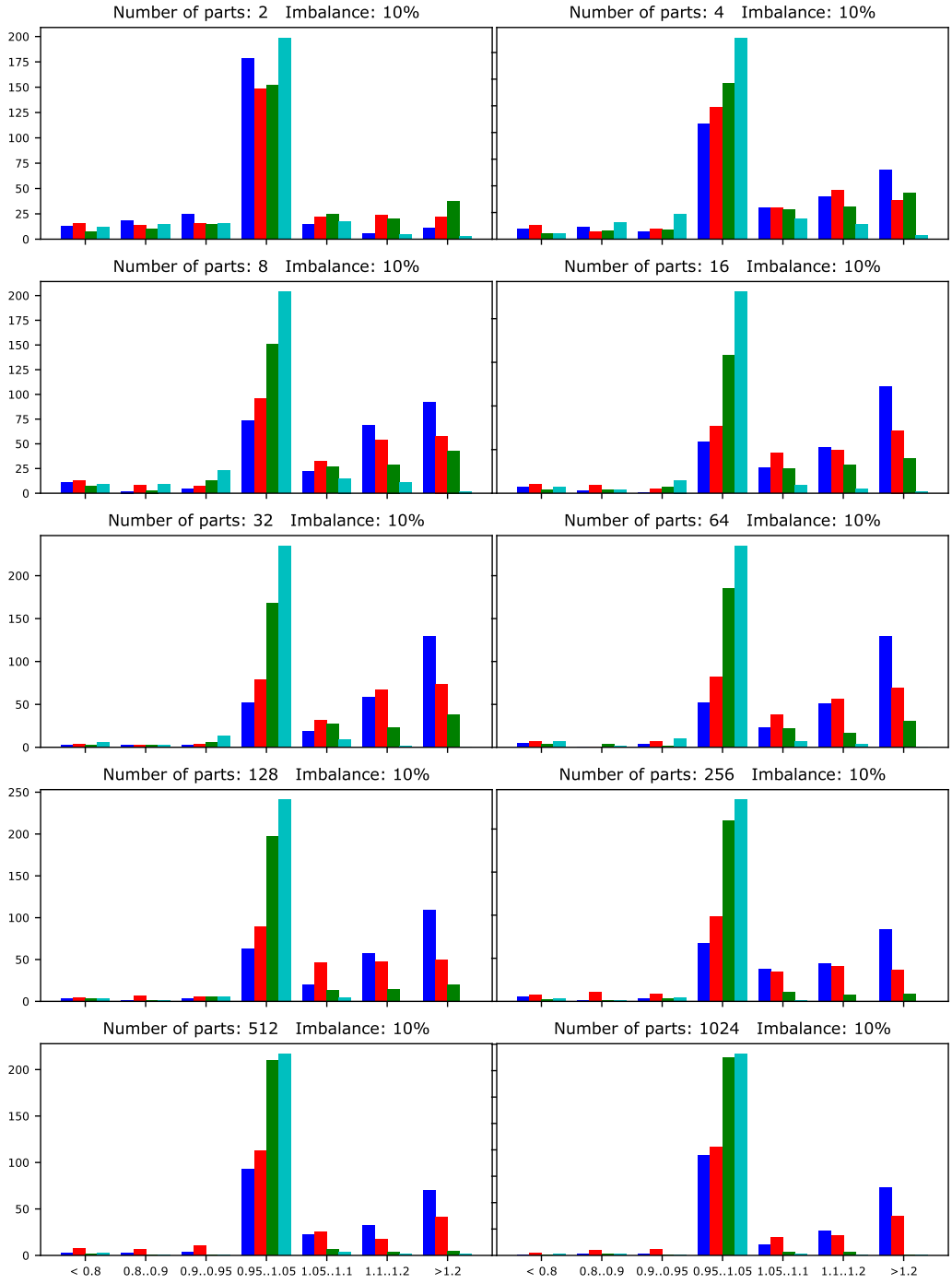


Figure 2.17: Histogram of ζ for coarsening using inner-product aggregation with additional parameters. Vertices are visited in the order of decreasing future volumes, connectivity metric is used. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

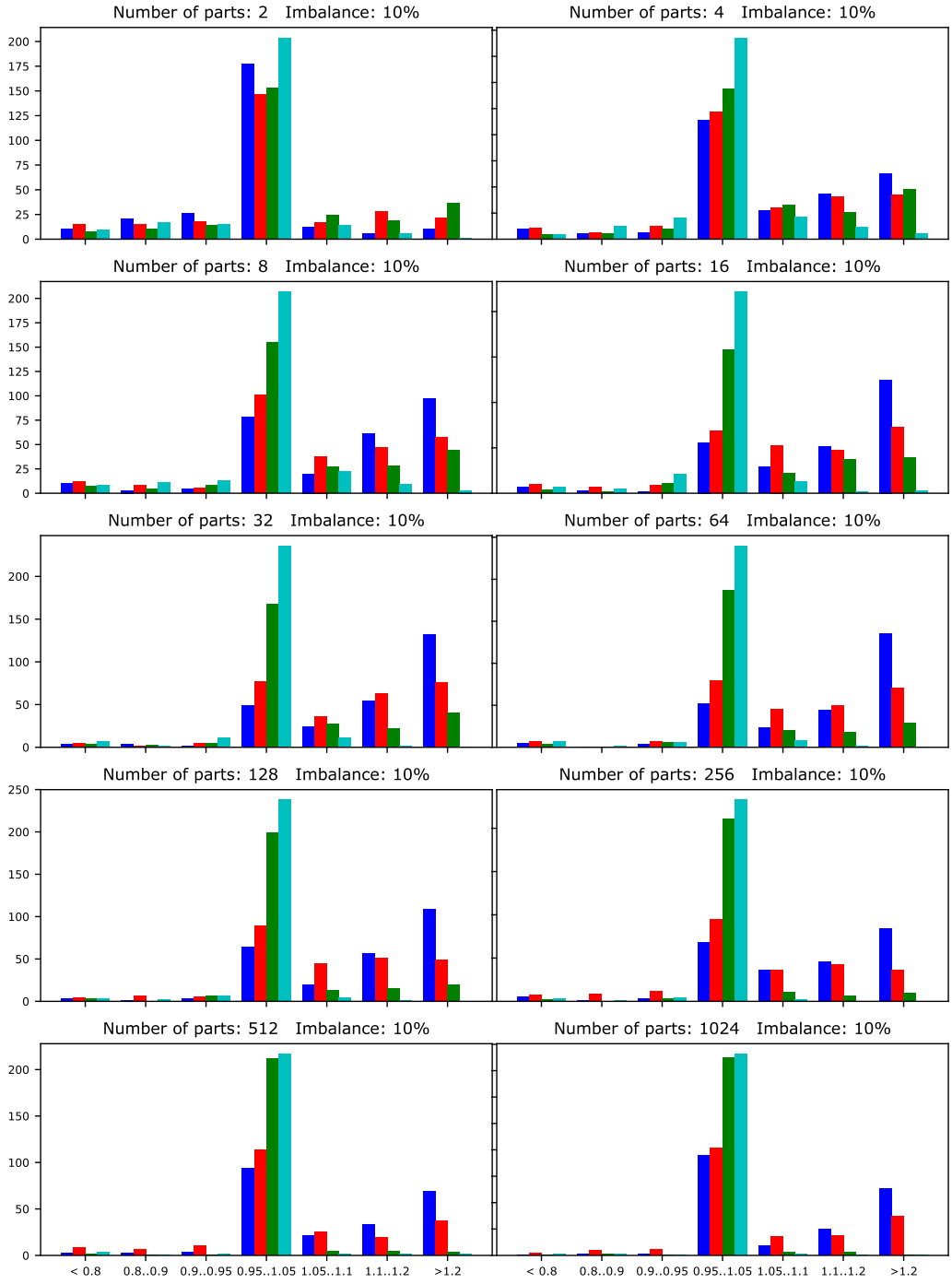


Figure 2.18: Histogram of ζ for coarsening using inner-product aggregation with additional parameters. Vertices are visited in random order, connectivity metric is used. Blue rectangle corresponds to PaToH, red to hMetis2, green to Zoltan PHG and cyan to Zoltan-AlgD

a significant improvement (average of 34.3%) over the same framework without algebraic distances, while decreasing the cut by more than two times for some hypergraphs. The algorithm is shown to outperform other state-of-the-art partitioners as well.

The experimental results indicate that one may gain substantial performance improvements through exploiting the global structure of highly irregular hypergraphs (e.g. social networks and other hypergraphs with power-law degree distribution). Exploiting the spectral properties of the hypergraph and its star expansion through some iterative procedure, like the one proposed in the work, is one way to achieve the gain. There remains ample room for improvement for current state-of-the-art hypergraph partitioners, particularly for the coarsening stage.

Chapter 3

Quantum Optimization

In recent years, quantum devices with up to tens of qubits on universal quantum computers (UQC) and a few thousand qubits on quantum annealer (QA) devices have become available. It enabled researchers to use real quantum hardware to solve "toy problems" for the first time. Unfortunately, in the near term, the devices are expected to stay very limited both in the number and the quality of qubits, making it hard to use these quantum computers for practical applications, which often require hundreds or even thousands of qubits. Such challenges as the qubit connectivity limitations, high level of noise, overhead of full error-correction, and concerns about scalability raise questions about the ability of near-term quantum hardware to effectively incorporate a larger number of qubits and deliver the theoretical speedups promised by many algorithms developed since 1990s

Hybridization of quantum and classical algorithms is one of the expedient answers that researchers suggest today to tackle real-life problems with existing quantum hardware. These hybrid algorithms combine both classical and quantum computers in an attempt to take the advantage of "the best of both worlds", leveraging the power of quantum computation while using a classical machine to address the limitations of Noisy Intermediate-Scale Quantum (NISQ) computers (see Fig. 3.1). This is true not only for optimization algorithms, as discussed in this work, but also for other problems, including quantum simulation [40], quantum machine learning [28, 203, 180] and more [154]. For example, classical computers have large memory and are capable of storing the entire global problem which is a challenge for NISQ devices with a small number of qubits. At the same time, quantum algorithms have shown improved performance for certain problems. To

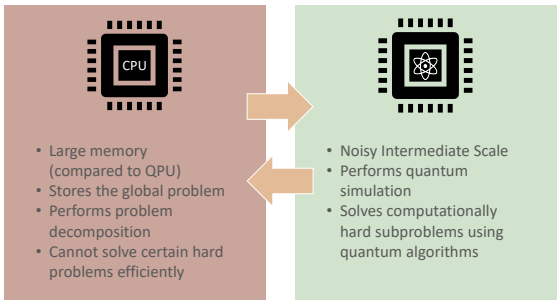


Figure 3.1: Hybrid algorithms combine computations performed in both, a classical computer (CPU) and a NISQ quantum computer (QPU). Hybrid algorithms are designed with the goal of leveraging the strength of each mode of computation while dealing with its weaknesses. For example, CPUs cannot efficiently perform quantum simulation whereas modern small near-term QPUs cannot compute problems with many variables.

unambiguously distinguish between the stages of computation performed on two principally different types of hardware, we will refer to the classical and quantum stages of hybrid algorithms as CPU (including such accelerators as GPUs and FPGAs) and QPU (Quantum Processing Unit, including quantum annealer and a universal quantum computer), respectively.

We primarily focus on two classes of NISQ-era devices, namely, UQCs and QAs, using IBM and D-Wave as exemplars. The IBM devices belong to the class of UQCs, which evolve the system by applying gates described using quantum assembly language (QASM). Other companies developing UQCs include Rigetti, Google, Microsoft, IonQ, and more. Alternatively, D-Wave devices are QAs designed to solve computational problems via quantum evolution towards the ground states of the cost Hamiltonians that encode quadratic integer optimization problems, without necessarily insisting on universality or adiabaticity[1]. While the two paradigms are very different, they share a lot of limitations and challenges.

3.1 Quantum Computing Paradigms

Quantum annealing (QA) can be considered a special, restricted case of adiabatic quantum computation (AQC) [157]. QA solves an optimization problem by encoding it as an Ising model Hamiltonian, with the ground state of that Hamiltonian corresponding to the global solution of the optimization problem. The Ising Hamiltonian describes the energy of a collection of n spin variables, with each variable being in one of two spin states (± 1). A spin configuration describes

[1] As a side note, noiseless adiabatic computation in general is universal and equivalent to gate-based model

assignment of states to spin variables, with s_i denoting the state of spin variable i (note that the 2-community problem maps naturally to this system, with the resulting spin state, s_i, denoting community assignment). The energy of a configuration is then defined by:

$$H(s) = \sum_{i>j} J_{ij} s_i s_j + \sum_i h_i s_i$$

where h_i correspond to external forces applied to spin variables, and J_{ij} to coupling strengths between pairwise spin interactions [157].

An equivalent mathematical formulation is the Quadratic Unconstrained Binary Optimization (QUBO) problem. The objective of a QUBO problem is to minimize (or maximize) the following function:

$$H(x) = \sum_{i<j} Q_{ij} x_i x_j + \sum_i Q_{ii} x_i, \quad x \in \{0,1\}.$$

QA finds the ground state of the objective Hamiltonian by performing a quantum evolution. As the initial Hamiltonian, QA uses a transverse field Hamiltonian. It introduces quantum fluctuations that help the annealing process to escape local minima by "tunneling through" hills in the energy landscape, enabling the evolution to move faster than adiabatic requirement would allow. As the evolution is performed, the transverse field Hamiltonian is slowly "turned off" (scaled with a coefficient decreasing to 0), such that the evolution finishes in a system described by the problem Hamiltonian [157].

Since AQC was introduced in 2000 by Farhi et al. [74], D-Wave Systems Inc [61], IARPA's QEO effort [117] and other researchers [179] have achieved a lot of progress in developing a system implementing QA [157] and applying it to a variety of problems, including optimization problems on graphs [245], machine learning [180], traffic flow optimization [175], integer factoring [190] and simulation problems [106]. Optimization problems can be solved by QA when formulated in the Ising form or as a quadratic binary optimization (QUBO).

Universal (or gate-based) quantum computing was introduced in the 1980s [66] and has seen great theoretical advances since. Shor's [231] and Grover's [93] algorithms are two most famous examples of quantum algorithms with theoretically proven speedups over classical state-of-the-art. Universal quantum computing has been implemented in hardware by a number of companies, na-

tional laboratories and universities [20, 23, 192, 7, 211].

To solve an optimization problem on a universal quantum computer, we use a hybrid quantum-classical approach, Quantum Approximate Optimization Algorithm (QAOA) [75, 76]. Similar to QA, a problem is encoded as an objective Hamiltonian H. Then a quantum evolution is performed starting with some fixed initial easy-to-prepare state (traditionally, uniform superposition over computational basis states is used). The difference is that unlike QA, in which the evolution is analog, in QAOA the evolution is performed by applying a series of gates parameterized by a vector of variational parameters θ. A hybrid approach, combining the quantum device performing the evolution and a classical optimizer, finds the optimal variational parameters. QAOA starts with an initial set of variational parameters θ_0. At each step, a multi-qubit state $|\psi(\theta)\rangle$ parameterized by the variational parameters θ is prepared on the quantum co-processor. Then a cost function $E(\theta) = \langle\psi(\theta)| H |\psi(\theta)\rangle$ is measured and the result is used by the classical optimizer to choose new parameters θ with the goal of finding the ground-state energy $E_G = \min_\theta \langle\psi(\theta)| H |\psi(\theta)\rangle$. QAOA provides a viable path to quantum advantage [73], making it a good algorithm to explore on near-term quantum computers.

QAOA has attracted considerable attention as a candidate algorithm for NISQ devices. When QAOA was originally introduced in 2014, it was shown to outperform the state-of-the-art classical solver for the combinatorial problem of bounded occurrence Max E3LIN2 [76]. (Thereafter, an improved classical algorithm was introduced that outperformed QAOA on this problem [21].) A recent paper [59] shows that QAOA (using a circuit with modest depth) can exceed the performance of Goemans-Williamson [90] algorithm for max-cut. In addition to these empirical results, theoretical results demonstrate that QAOA for max-cut improves on the best-known classical approximation algorithms for certain graphs [186, 256]. Although there is an active discussion about exactly how many qubits are required for meaningful quantum speedups [95, 226], the future of QAOA looks bright.

3.2 Quantum Approximate Optimization Algorithm (QAOA)

Consider a cost Hamiltonian $\hat{H}_C$ encoding the classical optimization problem (later in this section we present a cost Hamiltonian for network community detection and max-cut). Because the underlying optimization problem we are solving is *maximization*, we construct the cost Hamiltonian

$\hat{H}_C$ such that its highest-energy eigenstate encodes the solution, as opposed to the ground or lowest-energy state commonly used in VQE.[2] The goal of the hybrid algorithm is to prepare this eigenstate. In hybrid quantum-classical algorithms, the evolution is performed by applying a set of parameterized gates (ansatz). The goal then is to find a set of parameters that describe the evolution that prepares the desired state.

In QAOA, the quantum evolution starts in the initial state $|+\rangle^{\otimes n}$. Then the evolution is performed by applying two alternating operators based on the cost Hamiltonian $\hat{H}_C$ and mixing Hamiltonian $\hat{H}_M = \sum_i \hat{\sigma}_i^x$:

$$\begin{aligned} |\psi(\boldsymbol{\theta})\rangle &= |\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle \\ &= e^{-i\beta_p \hat{H}_M} e^{-i\gamma_p \hat{H}_C} \cdots e^{-i\beta_1 \hat{H}_M} e^{-i\gamma_1 \hat{H}_C} |+\rangle^{\otimes n} . \end{aligned} \tag{3.1}$$

Here p is the number of alternating operators or QAOA "steps." Then the objective function f (i.e., the energy of $\hat{H}_C$ in the state $|\psi(\boldsymbol{\beta}, \boldsymbol{\gamma})\rangle$) is

$$f(\boldsymbol{\beta}, \boldsymbol{\gamma}) = - \langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | \hat{H}_C | \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle . \tag{3.2}$$

Based on the value $f(\boldsymbol{\beta}, \boldsymbol{\gamma})$, the classical optimizer chooses the next set of parameters $\boldsymbol{\beta}, \boldsymbol{\gamma}$ with the goal of finding parameters that minimize f:

$$\begin{aligned} \boldsymbol{\beta}_*, \boldsymbol{\gamma}_* &= \arg\min_{\boldsymbol{\beta}, \boldsymbol{\gamma}} f(\boldsymbol{\beta}, \boldsymbol{\gamma}) \\ &= \arg\min_{\boldsymbol{\beta}, \boldsymbol{\gamma}} (- \langle \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) | \hat{H}_C | \psi(\boldsymbol{\beta}, \boldsymbol{\gamma}) \rangle). \end{aligned} \tag{3.3}$$

The objective function f is periodic with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$, allowing the parameters to be restricted to $\beta_i \in [0, \pi]$, $\gamma_i \in [0, 2\pi]$. Therefore the optimization domain is compact: $(\boldsymbol{\beta}, \boldsymbol{\gamma}) \in \mathcal{D} = ([0, \pi] \times [0, 2\pi])^p$.

3.3 Evaluating Quantum Approximate Optimization Algorithm: A Case Study

In this Section, we present a large-scale numerical study of the performance of QAOA on 90 random 10-node max-cut instances. For each problem instance we perform extensive (though not

[2]Note that in our case it is just a matter of convention, since introducing a minus sign changes a maximization problem into a minimization problem.

exhaustive) search of the variational parameter space, performing 990 million QAOA evaluations in total. We find that the average approximation ratio attained by QAOA on our set of problems is 0.77 as compared to the ground truth obtained by IBM CPLEX solver [56]. The maximum approximation ratio we observe is 0.91. We observe high variation in approximation ratios both between classes of instances and within the same class of instances. We observe that the difference between QAOA approximation ratio grows with the graph edit distance between underlying graphs. We find that QAOA parameters concentrate for the problem instances in our benchmark, indicating that the complexity of QAOA parameter optimization can be addressed by parameter reusing.

3.3.1 Problem Definition

We explore QAOA applied to graph maximum cut (or max-cut) problem. Consider a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. The goal of max-cut is to partition of the graph vertices V into two disjoint subsets V_1 and V_2, $V_1 \cup V_2 = V$, such that the total number of edges connecting the two subsets is maximized,

$$\max \|\{(u, v) \in E \text{ s.t. } u \in V_1, v \in V_2\}\| \tag{3.4}$$

(3.4) can be reformulated as [169],

$$\max_{\mathbf{s}} \sum_{i,j \in V} w_{ij} s_i s_j + c, \qquad s_k \in \{-1, 1\}, \forall k \tag{3.5}$$

where $w_{ij} = 1$ if $(i, j) \in E$ and 0 otherwise, and c is a constant. The binary decision variables s_i in (3.5) designate partition membership of the vertices of G after the cut. Finding an exact solution to the max-cut problem is known to be NP-hard [127]. To solve max-cut using QAOA, the cost Hamiltonian is constructed by mapping the binary variables s_k onto eigenvalues of Pauli Z operator $\hat{\sigma}^z$,

$$\hat{H}_C = \sum_{i,j \in V} w_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z. \tag{3.6}$$

Max-cut is the most well-studied target problem for QAOA due to the equivalence between max-cut and Unconstrained Quadratic Binary Optimization [199].

3.3.2 Methods

We follow Ref. [267] in performing extensive searches for QAOA parameters by running many instances of a relatively simple black-box local optimizer. We use derivative-free Bound Optimization BY Quadratic Approximation (BOBYQA) [195] as implemented in the NLopt nonlinear-optimization package [121]. BOBYQA was shown to perform well for QAOA parameter optimization [224] as compared to other off-the-shelf derivative-free optimization methods (see Section 3.4 for an in-depth discussion). We set the tolerances on change in the function value to 10^{-3} and on the change in optimization parameters to 10^{-2}. We allow BOBYQA 1 million evaluations for $p = 1, 2$ and 3 million for $p = 4, 6, 8$. BOBYQA is restarted from a new random point as it converges, with random starting points drawn from a uniform distribution over $\mathcal{D}$. In our experience, with the tolerance levels we use, BOBYQA takes 10-40 iterations to converge, resulting in 20,000-300,000 initial points (exceeding the 10,000 random initial points used in Ref. [267]).

Our benchmark consists of 90 Erdős-Rényi random graphs [71] with 10 nodes and edge creation probabilities e_p between 0.3 and 0.7 (10 random graphs for each value of e_p). We use high-performance quantum simulator Qiskit Aer [12] for noiseless simulations of QAOA circuits. We use NetworkX [101] for graph manipulations. We use GNU Parallel for large-scale numerical experiments [240]. All ensemble calculations were performed on Bebop cluster located in Argonne's Laboratory Computing Resource Center (LCRC) and Palmetto cluster at Clemson University.

3.3.3 Results

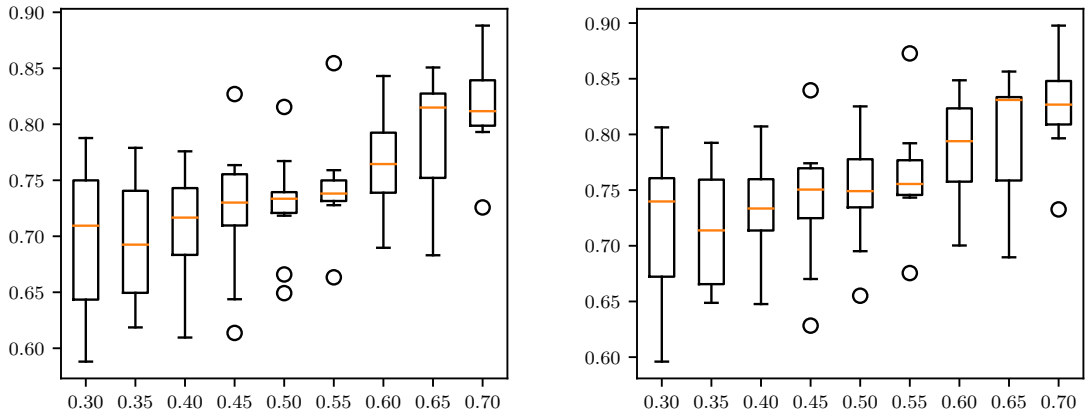
In this section we present the four main findings:

1. Optimization of QAOA parameters becomes challenging for derivative-free black box local optimizers even for relatively low number of steps p.

2. In low-depth regime, average approximation ratios attained by QAOA are limited (0.77 for our benchmark), making it challenging to compete with state-of-the-art classical heuristical solvers.

3. In low-depth regime, approximation ratios exhibit high degree of variability from one problem instance to another even within the same class of instances. The difference in approximation ratio grows with the graph edit distance between problem instances.

4. We observe strong concentration of optimal QAOA parameters, extending the results presented in Ref. [36]



(a) $p = 1$ (b) $p = 2$

Figure 3.2: Best approximation ratio found by QAOA for different problem classes as a function of edge creation probability e_p. (3.2a) presents results for $p = 1$ and (3.2b) for $p = 2$. (3.5a) presents the absolute difference in QAOA approximation ration as a function of graph edit distance.

First, we observe that despite considerable budget of evaluations ($1-3$ million evaluations, $\approx 100,000$ initial points) provided to classical optimizer, for $p > 4$ we do not obtain optimal variational parameters. As depth of QAOA is increased, the subspace reachable from the initial state is only increased. Therefore the best approximation ratio attained by QAOA with depth $p = k$ is always less or equal than the best approximation ratio for QAOA with depth $p = k + 1$.

However, we observe that for $p > 4$ the approximation ratios obtained by QAOA are lower than those for $p \leq 4$. We define approximation for problem instance G and a fixed QAOA depth p as the ratio between value of f given best found $\boldsymbol{\beta}, \boldsymbol{\gamma}$ and the solution obtained by CPLEX for the same problem instance:

$$r_{G,p} = \frac{f_{G,p}(\boldsymbol{\beta}_{\text{opt}}, \boldsymbol{\gamma}_{\text{opt}})}{\text{CPLEX}_G.}$$

As we allow CPLEX to converge to 0% gap, we are guaranteed that the CPLEX solution corresponds to the ground truth. Figure 3.3 presents this phenomenon on three representative problem instances. Figure 3.4 presents approximation ratios as a function of depth for the entire dataset. The median approximation ratio increases for $p = 1, 2, 4, 6$ and decreases for $p = 8$.

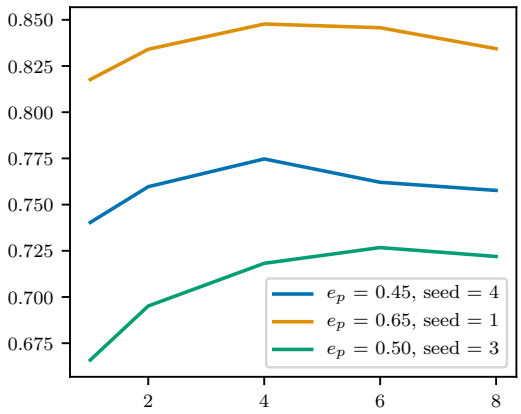


Figure 3.3: Three representative examples of approximation ratio decreasing with the number of QAOA steps due to suboptimality of the parameters $\boldsymbol{\beta}, \boldsymbol{\gamma}$. Three lines correspond to three Erdős-Rényi random graphs with 10 nodes and edge creation probabilities e_p. Seed corresponds to NetworkX implementation of Erdős-Rényi random graph generator [174].

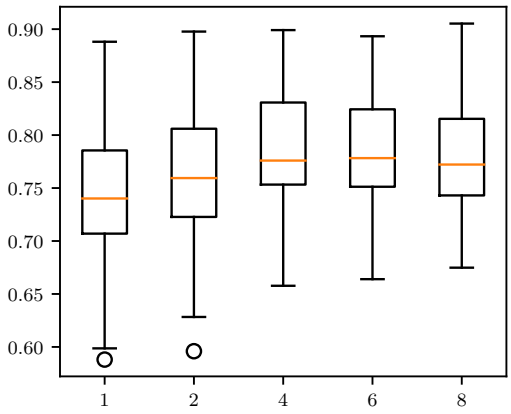


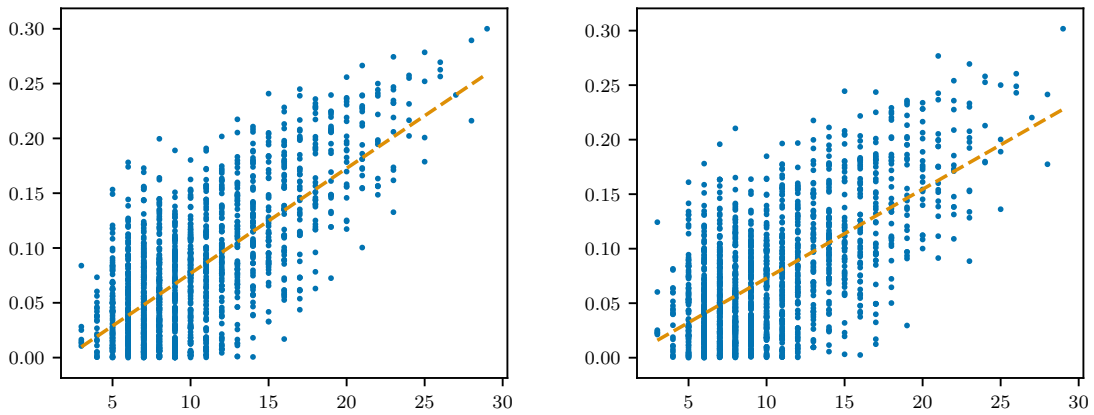
Figure 3.4: Boxplot of approximation ratio as a function of the number of QAOA steps p. Median approximation ratio increases for $p \leq 6$ and deacreases for $p = 8$. High variation of the approximation ratio attained by QAOA can be observed.

Second, we observe that the approximation ratios obtained by QAOA in low-depth regime are limited. Figure 3.2 presents the approximation ratios for different classes of problem instances for $p \in \{1, 2\}$. We follow [267] and examine the optimal parameters obtained for $p \in \{1, 2\}$ to confirm that they correspond to the global optimum of f. We observe that approximation ratio exhibits high variability within a problem instance class and does not exceed 0.91.

Third, we observe that the difference in approximation ratio obtained by QAOA grows with graph edit distance between problem instances (in other words, QAOA achieves similar approximation ratios for similar problems). Graph edit distance is a graph similarity measure. For graphs G_1 and G_2, graph edit distance is defined as minimum cost of edit path (a sequence of node and edge operations) transforming G_1 into a graph isomorphic to G_1. The absolute value of the difference in approximation ratio obtained by QAOA is defined as

$$d_{G_1,G_2} = |r_{G_1} - r_{G_2}|.$$

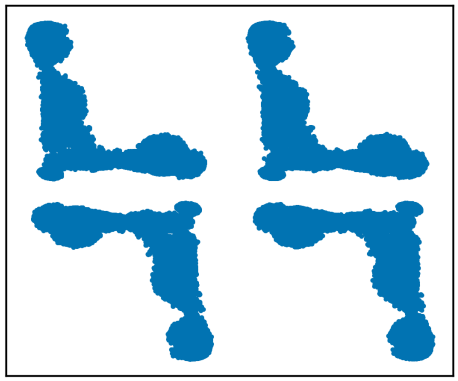
where $r_G = \max_{p\in\{1,2,4,6,8\}} r_{G,p}$. We present d as the function of graph edit distance in Figure 3.5.



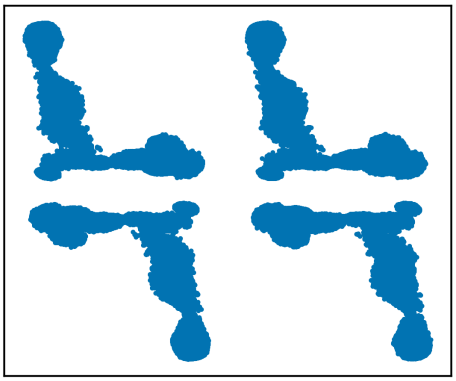
(a) $p = 1$ (b) $p = 2$

Figure 3.5: The absolute value of the difference in QAOA approximation ratio d as a function of graph edit distance between problem instances. Dashed trend line presents least squares linear fit.

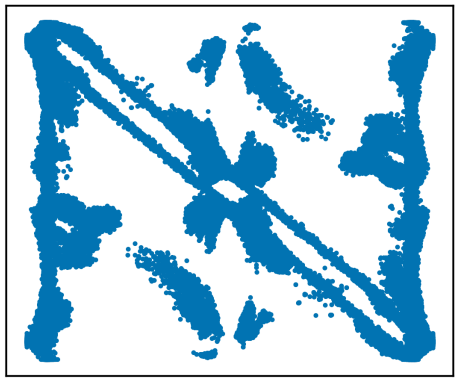
Fourth, we observe a strong concentration in optimal parameters. This has been observed previously for max-cut on 3-regular graphs [36]. Here we extend this observation to max-cut on Erdős-Rényi random graphs with unbounded vertex degree. Figure 3.6 presents the QAOA param-



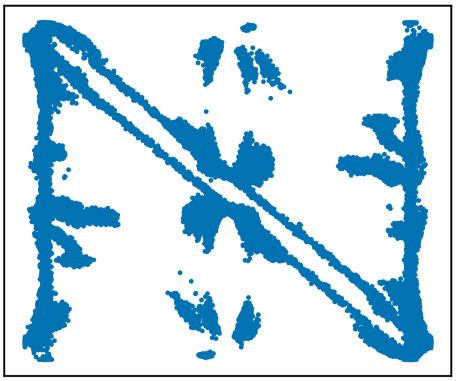
(a) $p = 1$



(b) $p = 1$, 50% of the points



(c) $p = 2$



(d) $p = 2$, 50% of the points

Figure 3.6: Parameters $\boldsymbol{\beta}, \boldsymbol{\gamma}$ corresponding to the values of approximation ratio within 1% of the best observed for a given problem instance. For $p = 2$ we only plot parameters corresponding to the second QAOA step.

eters $\boldsymbol{\beta}, \boldsymbol{\gamma}$ corresponding to the values of approximation ratio within 1% of the best observed for a given problem instance for the entire benchmark (i.e. all values of e_p). For $p = 2$ (Figure 3.6c and 3.6d) we present only the parameters corresponding to the second QAOA step. We demonstrate concentration by randomly removing points corresponding to 50% of instances. We observe that the optimal QAOA parameters concentrate around the same values for the problems in our benchmark.

3.3.4 Discussion

Our results highlight the need for further research into techniques for optimizing QAOA parameters and motivate the work presented in Section 3.4. It is clear that for to achieve good approximation ratios we need to go beyond $p = 1, 2, 4$. At the same time, as the depth increases, the limitations of local optimization methods become evident. The potential of QAOA cannot be realized without advances in variational parameter optimization, including through better understanding of the structure of QAOA objective.

Many recent results provide a path to scaling QAOA to higher p. FOURIER heuristic [267] is a promising approach, as it is shown to outperform brute force on some classes of max-cut problem instances. Combined with other approaches, like multistart methods [224] and gradient-based backpropagation-inspired approaches [59], these methods have the potential to make larger-depth QAOA competitive with classical state-of-the-art heuristics.

Finally, the concentration results presented in Figure 3.6 suggest that the QAOA training costs can be amortized across a class of problem instances. As optimal parameters concentrate around the same values, it should be possible to fit a model using precomputed optimal parameters for a subset of problem instances and then use that model to efficiently produce optimal QAOA parameters for other problem instances in that class. This approach, originally proposed in [36], can be combined with local optimization heuristics to further improve the performance.

3.4 Multistart Methods for Quantum Approximate Optimization

Variational hybrid quantum-classical algorithms like QAOA are considered the most promising path to demonstrating quantum advantage, that is, demonstrating superior performance of a

quantum system on some problem when compared with state-of-the-art classical methods. Demonstrating quantum advantage is a prerequisite for quantum computers to become a valuable high-performance computing resource. Variational hybrid algorithms, including VQE and variational implementations of QAOA, require reliable classical optimization methods to obtain solutions of good quality. Moreover, the performance of classical optimization methods in terms of the number of function evaluations directly translates into an improvement in performance of a variational quantum algorithm. Therefore, it is imperative that efficient and reliable optimization methods be developed for finding optimal variational parameters (see Section 3.3). Unfortunately, the parameter space for these problems is nonconvex and contains many low-quality, nondegenerate local optima [267]. Figure 3.7 shows an example energy landscape of a QAOA objective function with two parameters. This landscape has many low-quality optima that a local optimizer can get stuck in. In this Section, we address this challenge by using a multistart local optimization method. Our results are twofold. First, we explore direct optimization of QAOA parameters under realistic time constraints and show that the multistart framework APOSMM [138, 139] is able to find better parameters than single-start local search methods can (when using the same number of objective evaluations). Second, we demonstrate that the optimal QAOA parameters found for a given problem can be reused as an initial point for similar problems, both improving the quality of the solution and reducing the number of evaluations required to obtain it.

3.4.1 Problem Definition

We explore QAOA applied to the modularity maximization problem for the network community detection. Also known as graph clustering, network community detection aims to group vertices of the graph so that they are nontrivially connected compared with a random graph model. Modularity maximization often (but not necessarily) groups vertices so that there are as many edges as possible within the groups and as few as possible between the groups. Formally, for an undirected graph $G = (V, E)$ with two communities, modularity is defined as in [176]:

$$C = \frac{1}{4|E|} \sum_{ij} (A_{ij} - \frac{k_i k_j}{2|E|}) s_i s_j = \frac{1}{4|E|} \sum_{ij} B_{ij} s_i s_j, \tag{3.7}$$

where A is the adjacency matrix of G, k_i is the degree of vertex $i \in V$, and the variables $s_i \in \{-1, +1\}$ indicate community assignment of vertex i. That is, $s_i = -1$ denotes vertex i as being assigned

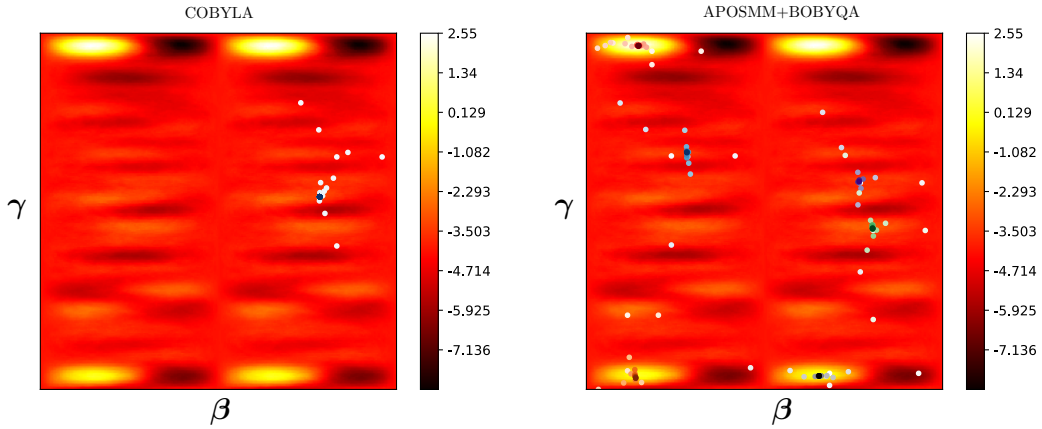


Figure 3.7: Energy landscape of QAOA objective function $\langle\psi(\beta,\gamma)|\,\hat{H}_C\,|\psi(\beta,\gamma)\rangle$ for modularity maximization community detection on connected caveman graph [257, 101] with 4 cliques of 4 vertices. Higher (white) is better. Left: the points evaluated by a single run of COBYLA [194, 196, 121]; each point corresponds to a pair (β,γ) that the local optimizer queried. Right: trace of APOSMM [138, 139] coordinating multiple COBYLA instances. Both methods were given a budget of 200 function evaluations.

to the first community, and $s_j = +1$ denotes that vertex j is assigned to the second community. Modularity maximization for general graphs is NP-hard [37] and has a variety of applications in complex systems [183, 236, 22, 178, 172].

The modularity maximization problem can be mapped onto QAOA by promoting variables s_i in (3.7) to Pauli spin operators $\hat{\sigma}^z$ [226, 222, 245], resulting in the Hamiltonian

$$\hat{H}_C = \frac{1}{4|E|}\sum_{ij} B_{ij}\hat{\sigma}_i^z\hat{\sigma}_j^z. \tag{3.8}$$

Multiple ansatzes (sets of gates used to produce trial state $|\psi(\boldsymbol{\theta})\rangle$) have been explored for QAOA, with the hardware-efficient ansatz [123] (originally proposed for VQE) being one of the most successful [226]. A similar ansatz leveraging nearest-neighbor interactions available on the device has been shown to achieve a better-than-random-guess approximation ratio for the max-cut problem on 3-regular graphs [77]. In this work we do not consider these ansatzes, however, because at the time of writing there is no evidence that QAOA with such ansatzes can beat the best classical algorithms. Instead, we focus on the alternating operator ansatz in (3.1).

3.4.2 Related Work

While the most commonly used strategy for identifying optimal QAOA parameters is using a classical optimizer in a variational loop, QAOA is not necessarily a variational algorithm. For example, Parekh et al. show that for one-step ($p = 1$) QAOA for max-cut on k-regular triangle-free graphs, parameters can be derived analytically [186]. Wang et al. show a similar result for one-dimensional antiferromagnetic rings [256]. More generally, Farhi et al. [75] proposed discretizing parameters into a grid. For N-qubit QAOA, however, this approach requires $N^{O(p)}$ evaluations, making it impractical even for small p. Finding good QAOA parameters remains a challenging problem, which motivates this work.

3.4.2.1 Parameter optimization in hybrid algorithms

Despite the recent advances in gradient-based methods [96, 27, 202, 267, 59, 107], gradient-free black-box methods remain the most common approach for optimizing parameters in hybrid quantum-classical algorithms. A variety of methods have been used, including the Nelder-Mead method [173] (for both QAOA parameter optimization [95, 96] and training quantum Boltzmann machines [249]), Bayesian methods [182], Powell's method [258], and an interior-point minimization method [264]. Researchers resort to derivative-free methods because analytic gradients for quantum circuits may not be available and approximating gradients can be computationally expensive [59]. (In some cases, algorithmic differentiation techniques may provide gradient information [27].) Since gradient-based methods can be sensitive to noise [268], they may be less suitable for noisy intermediate-scale quantum hardware.

A number of recent advances in finding good parameters have been made in the recent years, potentially making their optimization simpler. For QAOA, multiple results have shown connections between adiabatic schedule and QAOA parameters [36, 59, 267].

Zhou et al. [267] show that even at small depth p the schedule defined by optimal QAOA parameters is reminiscent of adiabatic quantum annealing, where $\hat{H}_C$ is gradually turned on while $\hat{H}_M$ is gradually turned off (see Sec. 3.4.1). Similar results were found by Crooks [59]. Additionally, Zhou et al. [267] demonstrate that the optimal values $\boldsymbol{\beta}_*, \boldsymbol{\gamma}_*$ have small variation between similar problem instances, a finding that we confirm in this work for a different graph problem (see Sec. 3.4.4). Zhou et al. use these insights to introduce a novel parameterization of QAOA and a heuristic optimization

scheme based on it.

Brandao et al. [36] show that for max-cut on 3-regular graphs, the objective function value is concentrated; that is, typical instances have nearly the same value of objective function. They make a case that the same holds for any combinatorial search problem where the number of clauses with a given variable is bounded (e.g., max-cut on a bounded-degree graph). They propose reusing optimal parameters between problems that come from the same distribution and refining them using a local optimization heuristic. In this work, we successfully apply this strategy to modularity clustering, a problem where the number of clauses in which a variable can appear grows with n (see Sec. 3.4.4).

Periodicity of the objective function with respect to QAOA parameters, visible on heatmaps in Fig. 3.7, has been demonstrated for max-cut [256, 267]. The periodicity was also observed for quasi-maximum-likelihood decoding of classical channel codes [153]. This can potentially allow for further restriction of the domain, eliminating some of the local optima and making the optimization problem easier. However, the theoretical results so far are problem specific. Therefore, we restrict our optimization domain to $\beta_i \in [0, \pi]$, $\gamma_i \in [0, 2\pi]$, following [75]. Note that this differs from the approach in [59], where the values of $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ were not constrained. A recent result shows that exploiting the periodicity of variational parameters of certain ansatzes for QAOA and other variational algorithms can improve optimization performance [168].

3.4.2.2 Derivative-free optimization methods

Selecting $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ values that maximize the objective function in (3.3) is a central optimization problem in variational algorithms. Since the gradient of the objective function with respect to $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ is unavailable on real quantum computers, researchers usually resort to so-called derivative-free optimization (DFO) methods: those that work only with observations of the objective function. Classical derivative-free direct-search methods are commonly applied to such problems: for example, Nelder-Mead is the default method for VQE problems in Grove [198]. Yet McClean et al. [154] shows that modern DFO methods achieve considerable benefits in terms of the number of function evaluations required. The BOBYQA method [195] is one such method for bound-constrained derivative-free optimization that builds quadratic models of the objective and optimizes them over a trust region in order to produce candidate points.

In the numerical optimization community, one commonly starts local optimization methods from different initial conditions in an attempt to identify better optima. While such an approach

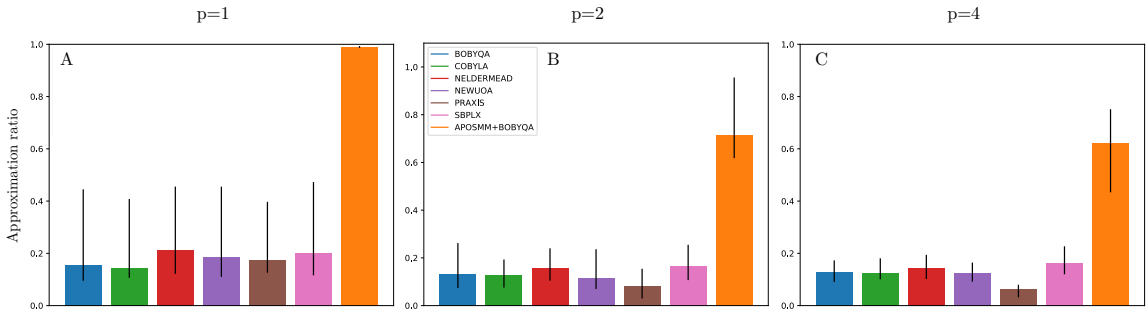


Figure 3.8: Ratio between the value of the objective function found by an optimization method and the best-found value. All local methods are run with *no* restart and zero tolerances. Heights of bars represent median over (10 seeds per problem) $\times$ (6 problems) = 60 runs. Error bars represent quartiles (25th and 75th percentiles). When compared with local methods without restarting, APOSMM finds solutions with much higher objective function values. This is due to local methods converging before exhausting the budget on number of function evaluations. p is number of QAOA steps ($p = 1$ corresponds to 2-dimensional domain $\mathcal{D}$ (A), $p = 2$ corresponds to $dim(\mathcal{D}) = 4$ (B), and $p = 4$ corresponds to $dim(\mathcal{D}) = 8$ (C).) Note that the approximation ratio 1.0 corresponds to the maximum value observed for a given problem and given value of p. The maximum absolute values of the objective function vary between the different numbers of QAOA steps.

may be easy to implement, it may result in unnecessary objective function evaluations. Assuming there are a finite number of local optima, the ideal approach would identify each using only a single local run.

The multilevel single linkage method (MLSL) [200, 201], uniformly samples points over the domain $\mathcal{D}$ and starts runs from those points that do not have a better point within a ball of some radius. They show favorable results for a specific approach for updating the radius as the number of sampled points increases, although such results are only asymptotic. MLSL was generalized by APOSMM [138, 139] to consider all points generated by an ensemble of local optimization runs, and not just those sampled from the domain.

3.4.3 Difficulty of Optimizing QAOA Parameters

In this section we present the results from using DFO methods to find optimal QAOA parameters. We use the high-performance simulator Qiskit Aer [12] to perform noiseless simulations of QAOA circuits. We measure the quality of the solution found by six derivative-free local optimization methods as implemented in the NLopt nonlinear-optimization package [121]: BOBYQA [195], COBYLA [194, 196], NEWUOA [193], Nelder-Mead [173], PRAXIS [42] and SBPLX [208]. We compare their performance to the implementation of APOSMM from the libEnsemble library [116].

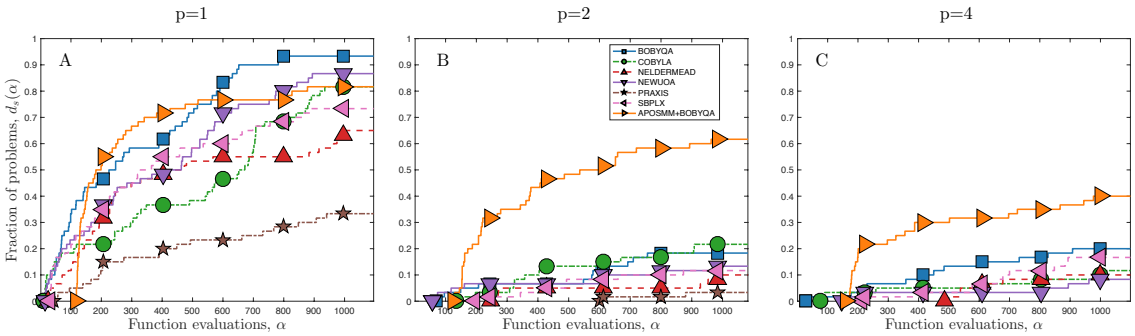


Figure 3.9: Data profiles for seven optimization methods on the $p = 1$ (A), $p = 2$ (B), and $p = 4$ (C) benchmark problems with $\tau = 0.01$. For the $p = 1$ (i.e., two-dimensional) problems, most methods are competitive; but as the number of parameters (i.e., circuit depth) increases, all methods have difficulty in identifying high-quality solutions on a large fraction of the test problems. Yet, we see that APOSMM+BOBYQA performs noticeably better.

APOSMM coordinates multiple local optimization runs in an attempt to identify better local optima. In this work, we use BOBYQA as the local optimization method within APOSMM (we denote this method as APOSMM+BOBYQA in figures). The performance of all methods is evaluated using two-way modularity maximization community detection problem on six synthetic graphs with community structure: three instances of connected caveman graph [257] and three instances of random partition graph [81]. All graphs have between 10 and 12 vertices and were generated with NetworkX [101]. The code used to perform the experiments is available on GitHub: `https://github.com/rsln-s/Multistart-Methods-for-Quantum-Approximate-Optimization`.

We performed two sets of experiments. First, we set the tolerances of local solvers to zero and allow them to run until convergence. The quality of the obtained solutions was then compared with the solutions found by APOSMM using the same number of evaluations. We observe that APOSMM finds solutions with a much higher value of the objective function (see Fig. 3.8). Since APOSMM is allowed another local optimization run after one has converged, a local method may not take full advantage of the function evaluations budget. To allow for a more equal comparison, we performed a second set of experiments. In the second set, we set the tolerances of local solvers to be equal to the tolerance of BOBYQA within APOSMM and if the method converges before exhausting function evaluations, it is restarted at a different random point. This restart scheme is essentially a naive version of MLSL. These results are also compared with APOSMM (see Fig. 3.9).

For both sets of the experiments, we limit the number of evaluations to 1,000. We choose this number as the realistic number of evaluations based on the estimates in [95]. We use the same

realistic if aggressive assumption of 1 millisecond per single run. Estimating the objective function in Eq. 3.2 requires thousands to tens of thousands measurements in practice [95, 123, 182]; we use an optimistic assumption of 1,000 measurements per run for obtaining the statistics to estimate the objective function. This gives an estimate on the time cost of performing the optimization equal to (time per single run) $\times$ (1,000 measurements per run) $\times$ (1,000 evaluations) $\approx$ 16 min. Note that this runtime is still orders of magnitude greater than the runtime of classical state-of-the-art MAXSAT solvers applied to the same problem [95]. Additionally, as the hardware is rapidly evolving, it is not possible to project these estimates into the future with certainty. However, it provides a useful estimate on the reasonable number of calls to the quantum device in a QAOA run.

Results show that a single run of a local optimization method cannot identify parameters $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ corresponding to a high-quality solution of the underlying problem (i.e., a high value of objective function). Fig. 3.8 shows that APOSMM is capable of finding parameters corresponding to values of objective function much larger than just the local solvers. This is partly due to local solvers converging before exhausting the limit on number of function evaluations (1,000).

If we set tolerances for local methods to the same values as in APOSMM (the tolerances on change in the function value to 10^{-3} and on the change in optimization parameters to 10^{-2}) and restart local methods after convergence, we observe that APOSMM is still solving more problems within the same budget of function evaluations. This is measured in the data profiles in Fig. 3.9; these data profiles track the fraction of problems solved to some level τ after a given number of function evaluations. Explicitly, if $t_{p,s}$ is the number of function evaluations required for each optimization method s to solve problem p in the set of problems P, then the data profile is

$$d_s(\alpha) = \frac{|\{p : t_{p,s} \leq \alpha\}|}{|P|}.$$

where α is the number of function evaluations. Data profiles require some definition of solving a problem to a level τ. For these problems, an optimization method s is determined to have solved problem p to a level τ after j evaluations if

$$f(x^0) - f(x^j) \geq (1 - \tau)(f(x^0) - \tilde{f}_p), \tag{3.9}$$

where x^0 is the problem's starting point, x^j is the jth point evaluated by the method, and $\tilde{f}_p$ is the

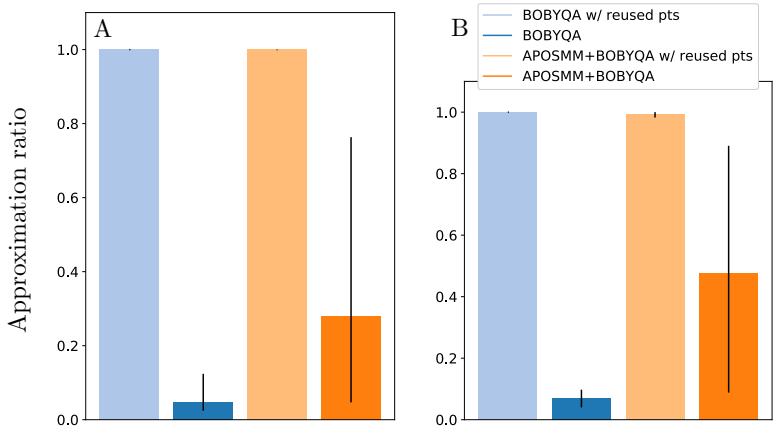


Figure 3.10: Ratio between the value of the objective function found by an optimization method and the best-found value. Left (A): we compare the best-performing local method and APOSMM with optimal points from similar problems ("w/ reused pts") and with random initial points. Heights of bars represent median over (10 seeds per problem) $\times$ (6 problems) $\times$ (5 different random edges removed) = 300 runs. Right (B): for each problem we remove only one "worst-case" edge. Error bars represent quartiles (25th and 75th percentiles). Reusing precomputed optimal points allows optimization methods to find better solutions (corresponding to higher objective values) within the same budget of function evaluations.

best-found function value by any optimization method on problem p. For example, if $\tau = 0.01$, the convergence test in (3.9) determines a method to solve problem p when a point is evaluated with 99% of the possible decrease on the problem (among the implementations being compared).

Figures 3.8 and 3.9 demonstrate that finding optimal parameters becomes increasingly harder as the dimension of the domain $\mathcal{D}$ (i.e., the number of QAOA steps p) increases. For $p = 1$, BOBYQA and APOSMM solve most of the problems (Fig. 3.9A) within 1,000 function evaluations, for $p = 2$ and $p = 4$ the best-performing method (APOSMM) solves only 60% and 40% of the problems, respectively. These results indicate that even for small number of QAOA steps ($p = 4$) direct optimization of variational parameters is hard under realistic time constraints.

3.4.4 Reusing Optimal QAOA Parameters

Sec. 3.4.3 presents results demonstrating the complexity of finding good QAOA parameters under realistic time constraints. Recently a number of researchers proposed amortizing the cost of finding good QAOA parameters for max-cut by reusing optimal parameters found for a given problem on similar problems [59, 267, 36]. We confirm and extend these findings by reusing optimal QAOA parameters found by exhaustive search. Optimal parameters for QAOA for modularity

maximization on a given graph are used as an initial guess for the local solver on a similar graph constructed by removing an edge from the original graph. This simulates a realistic scenario of solving community detection on a dynamical graph, for example, a social network where new friend connections are dynamically added and removed.

We estimate true optimal parameters by setting the tolerance on the change in the function value to 10^{-3} and the tolerances on the change in the parameters to 10^{-2} and restarting BOBYQA after each convergence until 100,000 function evaluations have been used. We observe that this exhaustive approach identifies multiple high-quality local optima. We then use these high-quality QAOA parameters as initial guesses for local methods and APOSMM. After a local method converges, it is restarted from the next-best local optima.

Our contribution extends previous work in two ways. First, we consider a different optimization problem, namely, modularity community detection. Second, in addition to random similar problems, we consider "worst-case" small changes. To simulate a "worst-case" scenario, we remove an edge from the graph that has the greatest impact on its spectrum. Concretely, we compute the spectrum of the graph Laplacian before and after removing an edge. The change in the spectrum is measured by computing the Euclidean distance between the eigenvectors of the graph Laplacians. The graph spectrum has deep connections to many optimization problems on graphs, including graph partitioning and community detection [54, 167].

Figure 3.10 presents the results. We observe that using optimal parameters from similar problems allows optimization methods to find high-quality solutions under realistic time constraints. Thus, we are hopeful that the high cost of finding good QAOA parameters can be amortized by reusing the parameters from similar problems.

3.4.5 Discussion

This section present results on finding optimal QAOA parameters to improve the performance of quantum optimization solvers. We show that multistart methods such as APOSMM can utilize a fixed number of function evaluations more efficiently by interleaving multiple local optimization runs and considering all $(\boldsymbol{\beta}, \boldsymbol{\gamma})$ parameters generated by them. We observe that as the number of QAOA steps and the dimension of the corresponding optimization domain $\mathcal{D}$ is increased, the optimization problem becomes increasingly hard. These results highlight the need to develop more efficient approaches to finding optimal parameters to accelerate and improve the performance of

QAOA—a challenge because, in order to compete with state-of-the-art classical solvers on problems with fewer than 200 variables, QAOA has to run in no more than a minute [95, 226]. An additional challenge is presented by the high levels of noise on near-term hardware.

We show that the obstacles can be partially addressed by reusing optimal parameters found for a similar problem. We observe that parameters can be reused both for similar problems with a random change introduced and in "worst-case" scenarios, where the change in the underlying problem has the greatest impact on its structure. For example, reusing optimal parameters found for $p = 1$ using BOBYQA or APOSMM for a dynamic graph over 1,000 changes and allowing local methods a realistic 10–30 iterations in order to refine reused optimal points at each iteration, would bring amortized cost down from (1 ms per run) $\times$ (1,000 measurements per run) $\times$ (1,000 evaluations) $\approx$ 16 minutes to a more competitive $\approx$ 10 seconds. Reusing parameters and employing heuristic techniques such as FOURIER proposed in [267] could bring down amortized costs of a QAOA run even further. We believe this could make quantum optimization solvers a valuable extreme-computing resource.

The limited connectivity between qubits in many hardware implementations presents an additional challenge. For example, superconducting qubit technology, developed by, among others, IBM, Rigetti, and Google, provides only nearest-neighbor connectivity with qubits arranged on a two-dimensional lattice. The modularity maximization graph clustering problem discussed in this Section requires all-to-all connectivity. The connectivity limitation can be addressed by a SWAP network [15, 18, 59] with only $O(N)$ overhead (where N is the number of qubits). Additionally, ion-trap architectures (the most famous implementation developed by IonQ) do not have the same connectivity limitations, because they allow the application of gates between any pair of qubits.

All these factors strengthen the potential of QAOA. As hardware continues to improve and more advanced techniques for parameter optimization are developed, QAOA has the potential to outperform classical state-of-the-art solvers.

3.5 Subspace Symmetry Predicts QAOA Performance

A number of recent results have shed some light on the potential of QAOA as compared with classical algorithms. Nontrivial performance guarantees for QAOA circuits have thus far been obtained only in very limited settings, e.g., [75, 256, 98, 119]. A recent paper of Hastings [109] demonstrates that for QAOA with (high-level circuit depth) $p = 1$, classical local algorithms can

achieve the same (or better) performance guarantees as QAOA, and shows that for some problems increasing the QAOA depth by any bounded amount would not lead to significant improvements in performance. Bravyi et al. [41] show that ground states of some families of Hamiltonians with $\mathbb{Z}_2$ symmetry (exhibited, for example, by the MaxCut problem) cannot be prepared with constant depth quantum circuits. A corollary of this result shows that constant-depth QAOA for the Max-Cut problem is outperformed by the classical Goemans-Williamson algorithm for some families of graphs. Farhi et al. [78] show an upper bound for the approximation ratio attainable by a particular realization of QAOA for MaxIndependentSet on d-regular graphs with depth growing less than logarithmically with number of nodes, implying that the depth p must grow at least logarithmically for QAOA to outperform a certian approximation ratio. They note the relationship between the required depth and the graph diameter, suggesting that for graphs with larger diameter (e.g., 2d lattices) the depth required to obtain a quantum advantage may be polynomial in the problem size in this case.

While clearly important, these findings leave many important practical questions unresolved, for instance, "What is the QAOA depth required to adequately solve a given problem?" For example, Farhi et al. [78] note that a class of 3-regular graphs with 2 million qubits, the upper bound they prove only yield a necessary depth of $p = 7$. Similarly, the majority of rigorous bounds to the optimal approximation ratio achieved by QAOA have been obtained only for small depth, primarily $p = 1$ [75, 76, 256, 98]. Furthermore, such results are typically worst-case and do not take advantage of the structure of a given problem instance. Indeed, initial numerical studies on small problem instances have indicated that going beyond small p appears necessary to obtain potential quantum advantage [220, 256, 267, 59].

Our work aims to help bridge the gaps between existing performance bounds and numerical results by introducing a novel approach to studying QAOA and the mechanism by which it explores qubit Hilbert space (i.e., the space of possible solutions). Our approach leverages connections between the quantum symmetry of the QAOA state and operators, and the classical symmetries present in the underlying optimization problem instance. The primary example we consider throughout the Section is the MaxCut problem, for which the classical symmetries we consider are the permutations of the problem graph (i.e., graph automorphisms). The MaxCut problem has been studied for QAOA in [75, 256, 98, 59, 267, 220, 95, 41], among others.

Our contributions are two-fold. In the first part (Section 3.5.2), we prove a series of results

connecting the symmetry of QAOA reachable space to the symmetry of the (hyper)graph representation of the optimization problem in the general case. Our results cover a wide range of problems, including general quadratic unconstrained binary optimization (QUBO) and Ising form problems and can be extended to higher-order problems by generalizing graphs to hypergraphs (Section 3.5.2.1). Our theoretical results are applicable both to transverse field mixer QAOA introduced by Farhi [75], as well as to its generalization, Quantum Alternating Operator Ansatz [100]. In the second part (Section 3.5.3), we use this connection to study QAOA performance. While our theoretical results are general and impose no constraints on QAOA schedules, in Section 3.5.3 we limit QAOA to linear schedules to sidestep the challenge of finding optimal QAOA parameters (Section 3.5.3.1) and focus specifically on the MaxCut problem. We study MaxCut as an application because its classical symmetries are easily understood as those of the underlying graph for each problem instance. We use exact and approximate graph symmetry measures (defined in Section 3.5.3.2) as features for machine learning models that we train to predict QAOA performance as defined by the minimum depth required to achieve a given approximation ratio on the target problem. We discuss our choice of target approximation ratio for MaxCut in depth in Section 3.5.3.3. We use Support Vector Regression and an ensemble of Support Vector Machine classifiers to achieve median absolute error on predicted depth of < 1.5.

Reproducibility: We make the code for the experiments in Section 3.5.3 and the entire generated dataset available online [1, 5].

3.5.1 Background

In this section we briefly review our notions of binary optimizaiton, QAOA, and classical symmetries. Consider a function $f(x)$, $x \in \{0,1\}^n$, and a binary optimization problem

$$\max_x f(x), \qquad x \in \{0,1\}^n, \tag{3.10}$$

which may be encoded as the n-qubit Hamiltonian

$$C = diag(f(x)), \qquad C \in \mathbb{C}^{2^n \times 2^n}.$$

A compact representation of such Hamiltonians (in terms of Ising spin or Pauli matrices) can be

constructed efficiently for many combinatorial optimization problems [99]. A binary string $\mathbf{x}^* \in \{0,1\}^n$ achieves approximation ratio $r \leq 1$ if

$$\frac{f(\mathbf{x}^*)}{\max_x f(x)} \geq r,$$

and an algorithm is said to give an r-approximation for (3.10) if it returns a string achieving at least r for every problem instance (i.e., in the worst case).

The particular optimization problem that we focus on in this Section is MaxCut on unweighted graphs. Given a graph $G = (V, E)$, the goal of the problem is to partition the set of nodes v into two disjoint parts such that the number of edges with endpoints in both parts is maximized. The set of edges that span both parts is called a cut. The MaxCut problem is APX-complete [185], which means it has no polynomial-time approximation scheme in general unless P=NP (i.e., it cannot be approximated in polynomial time better than some constant), but efficient approximation algorithms exist for particular classes of graphs [16]. MaxCut is encoded by Hamiltonian $C_{\text{MaxCut}} = \frac{1}{2}\sum_{(u,v)\in E}(I - Z_u Z_v)$, where Z_j is a single-qubit Pauli-Z operator applied to qubit u.

Quantum Approximate Optimization Algorithm (QAOA) is a hybrid quantum-classical algorithm that combines a parameterized quantum evolution with a classical outer-loop optimizer to approximate the ground state of the problem or cost Hamiltonian C. At each call to the quantum computer, a trial state is prepared by applying a series of quantum alternating operators:

$$|\vec{\beta}, \vec{\gamma}\rangle_p = |\vec{\beta}, \vec{\gamma}\rangle = U_B(\beta_p)U_C(\gamma_p)\ldots U_B(\beta_1)U_C(\gamma_1)\,|\psi\rangle\,, \tag{3.11}$$

where $U_C(\gamma) = e^{-i\gamma C}$ is the phase operator, $U_B(\beta)$ is the mixing operator and $|\psi\rangle$ is some easy to prepare initial state. We refer to the number of alternating operator pairs p as the *QAOA depth*. The parameters $\vec{\beta}, \vec{\gamma}$ are said to define a *schedule*, analogous to the choice of a schedule in quantum annealing. For unconstrained optimization problems that we consider in this Section, the mixing operator is $U_B(\beta) = e^{-i\beta\sum_j X_j}$ (i.e. time evolution under the transverse field Hamiltonian $B = \sum_j X_j$), and the initial state is the uniform superposition $|\psi\rangle = |+\rangle^n$. Here X_j is single-qubit Pauli-X operator applied to qubit j. We let $Pr_p(x$ denote the probability of obtaining bitstring $x \in \{0,1\}^n$ when measuring the QAOA state:

$$Pr_p(x) = |\langle x|\vec{\beta},\vec{\gamma}\rangle_p|^2.$$

For given phase and mixing operators and initial state, we define the *QAOA reachable space* or *QAOA subspace* to be the space of states exactly reachable by QAOA with any schedule and depth:

$$|\vec{\beta},\vec{\gamma}\rangle_p = |\vec{\beta},\vec{\gamma}\rangle_{p\in\mathbb{N},\vec{\beta},\vec{\gamma}\in\mathbb{R}^{2p}}.$$

Consider the group S_{2^n} of permutations on n bit strings (i.e., permutations of 2^n objects), and its natural subgroup $S_n \subset S_{2^n}$ of permutations of the bit indices. We define a *classical symmetry* of the cost function $f(x)$ to be a transformation $\sigma \in S_n : \{0,1\}^n \to \{0,1\}^n$ such that for all $x \in \{0,1\}^n$ we have $f(x) = f(\sigma(x))$. We may lift $\sigma(x)$ to the quantum operator $A = A(\sigma) = \sum_{\vec{x}\in\{0,1\}^n} |x_{\sigma(1)}\dots x_{\sigma(n)}\rangle\langle x_1\dots x_n|$ induced by σ, which acts as $A|x\rangle = |\sigma(x)\rangle$ for each computational basis state $|x\rangle$. Equivalently, we may write $A \in S_{2^n}$, where here the representation of S_{2^n} is implicitly understood.

A state $|\phi\rangle$ is said to be symmetric under operator A if it is invariant under A, i.e., $A|\phi\rangle = |\phi\rangle$. Equivalently, $|\phi\rangle$ is symmetric under A if its projection onto symmetric components is preserved: $\langle x|\phi\rangle = \langle x|A|\phi\rangle = \langle a(x)|\phi\rangle$ for all $x \in \{0,1\}^n$. In particular, observe that for any $A \in S_{2^n}$ the QAOA initial state $|s\rangle = |+\rangle^n$ satisfies $A|s\rangle = |s\rangle$. A subspace is said to be symmetric under an operator A if each state in this subspace is symmetric under A.

An illustrative example is $\mathbb{Z}_2$ symmetry for QAOA state for MaxCut problem. Note that both MaxCut and transverse field Hamiltonians commute with $A = X_1X_2\dots X_n$. Therefore it is easy to see that both phase and mixing operators will affect $|x\rangle$ and $A|x\rangle = |\neg\mathbf{x}\rangle$ components of QAOA state equally. Therefore $\langle x|\vec{\beta},\vec{\gamma}\rangle = \langle\neg\mathbf{x}|\vec{\beta},\vec{\gamma}\rangle, \forall x \in \{0,1\}^n$ (see Theorem 3.5.1 and Corollary 3.5.2.4 for more general discussion). In particular, observe that this symmetry satisfies $A \in S_{2^n}$ but $A \notin S_n$.

Similarly to the symmetry of QAOA subspace, graph symmetry is a transformation of a graph that preserves its structure. The most well-studied such transformation is a (vertex) automorphism. Graph automorphism is a permutation of nodes that preserves edges. More formally, for a graph $G = (V, E)$, automorphism is a permutation $\sigma : V \to V$ s.t. $(\sigma(v), \sigma(u)) \in E$ iff $(u, v) \in E$. Two vertices u, and v are said to belong to the same orbit if there exists an automorphism σ such that $\sigma(v) = u$. Note that orbits impose classes of equivalence on the graph G. For a further discussion of graph automorphisms and their properties the reader is referred to the third part of [32] by

Biggs.

3.5.2 Graph theoretical approach for analyzing QAOA symmetry

We establish a novel connection between the symmetry of QAOA subspace and the symmetry in the graph representation of the problem. We begin by showing that if an operator commutes with both mixing and cost Hamiltonians and preserves the initial state, then it preserves QAOA (in the sense of Theorem 3.5.1 below). Then we show that graph automorphism satisfies this condition for QAOA for unconstrained problems (transverse field mixing operator and uniform superposition initial state). Finally, we explicitly show how the graph representations can be constructed for common problem classes (Corollaries 3.5.2.4 and 3.5.2.5).

Lemma 1. *Consider QAOA of depth p with problem Hamiltonian C, mixing Hamiltonian B and initial state $|\psi\rangle$:*

$$|\vec{\beta},\vec{\gamma}\rangle_p = |\vec{\beta},\vec{\gamma}\rangle = U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)\,|\psi\rangle$$

Consider $A \in S_{2^n}$ such that:

1. $[A, U_C(\theta)] = 0,\ \forall\theta \in \mathbb{R}$
2. $[A, U_B(\theta)] = 0,\ \forall\theta \in \mathbb{R}$
3. $A^\dagger\,|\psi\rangle = |\psi\rangle$

Then QAOA is symmetric under A, i.e., if $A\,|x\rangle = |a(x)\rangle$, with $x, a(x) \in \{0,1\}^n$, then $|x\rangle$ and $|a(x)\rangle$ will have the same overlap with QAOA state

$$\langle x|\vec{\beta},\vec{\gamma}\rangle = \langle a(x)|\vec{\beta},\vec{\gamma}\rangle$$

for all such $\vec{\gamma},\vec{\beta}$.

Proof.

$$\begin{aligned}\langle a(x)|\vec{\beta},\vec{\gamma}\rangle &= \langle a(x)|\, U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)\,|\psi\rangle \\ &= \langle a|\, S^\dagger U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)\,|\psi\rangle \\ &= \langle a|\, U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)S^\dagger\,|\psi\rangle \\ &= \langle a|\, U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)\,|\psi\rangle \\ &= \langle a|\vec{\beta},\vec{\gamma}\rangle\end{aligned}$$

□

In practice, we are interested not in overlap between states, but rather in the probability of observing a given measurement outcome. The results of Lemma 1 can be easily reformulated in terms of probabilities.

Lemma 2. *Consider QAOA of depth p with problem Hamiltonian C, mixing Hamiltonian B and initial state $|\psi\rangle$ and a symmetry $A \in S_{2^n}, A\,|x\rangle = |a(x)\rangle$ satisfying conditions 1-3 of Lemma 1.*

Then applying A does not change the probability of obtaining a given bitstring x when measuring the QAOA state: $\forall x \in \{0,1\}^n$, $Pr_p(x) = Pr_p(a(x))$

Proof. From Lemma 1 we have $\langle x|\vec{\beta},\vec{\gamma}\rangle = \langle a(x)|\vec{\beta},\vec{\gamma}\rangle$. Therefore:

$$Pr_p(a(x)) = |\,\langle a(x)|\vec{\beta},\vec{\gamma}\rangle\,|^2 = |\,\langle x|\vec{\beta},\vec{\gamma}\rangle\,|^2 = Pr_p(x)$$

□

The conditions of Lemma 1 are general, but are hard to verify in practice. Therefore in many cases a formulation focusing on cost and mixer Hamiltonians, rather than operators, is more natural.

Theorem 3.5.1. *Consider QAOA of depth p with problem Hamiltonian C, mixing Hamiltonian B and initial state $|\psi\rangle$:*

$$|\vec{\beta},\vec{\gamma}\rangle_p = |\vec{\beta},\vec{\gamma}\rangle = U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)\,|\psi\rangle$$

Consider $A \in S_{2^n}$ such that:

1. $[A, C] = 0$

2. $[A, B] = 0$

3. $A^\dagger |\psi\rangle = |\psi\rangle$

Then applying A does not change the probability of obtaining a given bitstring x when measuring the QAOA state: $\forall x \in \{0,1\}^n$, $Pr_p(x) = Pr_p(a(x))$

Proof. If $[A, C] = 0$ then $[A, U_C(\theta)] = 0 \ \forall \theta \in \mathbb{R}$, and if $U_B(\theta) = e^{-i\theta B}$ and $[A, B] = 0$ then $[A, U_B(\theta)] = 0 \ \forall \theta \in \mathbb{R}$. Therefore A satisfies the conditions of Lemma 2, which completes the proof. □

Remark. *In the conditions for Lemma 1, the angles θ need not extend over all reals. Generally, for a particular application it may be beneficial to restrict the range of angles considered and only require the symmetry to hold for that range. Note that this is a practically interesting restriction that is weaker than $[A, C] = 0 = [A, B]$. However, in this Section we consider the simpler, general case.*

Theorem 3.5.2 (QAOA symmetry group)**.** *Consider QAOA of depth p with problem Hamiltonian C, mixing Hamiltonian B and initial state $|\psi\rangle$:*

$$|\vec{\beta}, \vec{\gamma}\rangle_p = |\vec{\beta}, \vec{\gamma}\rangle = U_B(\beta_p)U_C(\gamma_p)\dots U_B(\beta_1)U_C(\gamma_1)|\psi\rangle$$

Consider $A_1, \dots A_\ell \in S_{2^n}$ such that:

1. $[A_j, C] = 0$

2. $[A_j, B] = 0$

3. $A_j^\dagger |\psi\rangle = |\psi\rangle$

Then taking matrix products of the A_j generates a subgroup $\mathcal{A} = \{A_0 = I, A_1, \dots A_\ell, A_{\ell+1} \dots A_{\ell'}\} \subset S_{2^n}$ of order $\ell' \geq \ell$ such that QAOA is symmetric under A_j, $j = 1, \dots, \ell'$, i.e., if $A_j |x\rangle = |a_j(x)\rangle, x, a_j(x) \in \{0,1\}^n$, then the bitstrings x and $a_j(x)$ will have the same probability of being found upon measuring the QAOA state: $Pr_p(x) = Pr_p(a_j(x))$.

We call this subgroup $\mathcal{A} \subset S_{2^n}$ ***QAOA symmetry group****.*

Proof. Follows trivially from repeating proof of Theorem 3.5.1 for each $A = A_j$. □

Corollary 3.5.2.1. *Consider an optimization problem with objective function $f(x), x \in \{0,1\}^n$. If $\sigma \in S_n$ is a permutation of variables (qubits) that preserves the objective f ($f(x) = f(\sigma(x))$, $\forall x \in \{0,1\}^n$), $B = \sum_i X_i$ is the transverse field Hamiltonian and initial state is uniform superposition $|\psi\rangle = |+\rangle^n$, then QAOA is symmetric under permutation operator $A(\sigma)$ induced by σ, $A \in S_{2^n}$, i.e., if $A|x\rangle = |\sigma(x)\rangle, x, \sigma(x) \in \{0,1\}^n$, then the bitstrings x and $\sigma(x)$ will have the same probability of being found upon measuring the QAOA state: $Pr_p(x) = Pr_p(\sigma(x))$.*

Proof. Let's show that if $\sigma \in S_n$ preserves the objective f, then $A = A(\sigma) \in S_{2^n}$ satisfies the conditions of Theorem 3.5.1. Condition (1) is satisfied trivially, since $C = diag(f(x))$, so if σ preserves the objective f, then A will only permute elements that correspond to the same values of the f: $AC = CA = C$. Since $B\sum_j X_j$, B acts on all qubits in the same way, so $AB = BA$ (i.e., it doesn't matter if we first permute the qubits and apply B or first apply B and *then* permute the qubits) and Condition (2) is satisfied. Condition (3) is satisfied trivially. □

Corollary 3.5.2.2. *Consider graph G, $|V| = n$ with some (label-independent) objective function f defined on G. If $\sigma \in S_n$ is an automorphism of G, $B = \sum_i X_i$ is the transverse field Hamiltonian and initial state is uniform superposition $|\psi\rangle = |+\rangle^n$, then QAOA is symmetric under $A(\sigma)$.*

Proof. Since a graph automorphism preserves graph G, it preserves any label-independent objective defined on graph G. Combined with Corollary 3.5.2.1 this concludes the proof. □

Corollary 3.5.2.3 (Graph automorphism group is a subgroup of QAOA symmetry group)**.** *Consider graph G, $|V| = n$, its group of automorphisms $Aut(G) \subset S_n$ and some (label-independent) objective function f defined on G. Consider QAOA with transverse field mixing operator $B = \sum_i X_i$ and initial state $|\psi\rangle = |+\rangle^n$, and the group of permutations operators $\mathcal{A}(G) \subset S_{2^n}$ induced by elements of $Aug(G) \subset S_n$. Then $\mathcal{A}(G)$ is a subgroup of QAOA symmetry group.*

Proof. Follows trivially from repeating Corollary 3.5.2.2 for each element of $\mathcal{A}(G)$. □

Remark. *In practice obtaining the full symmetry group is often prohibitively expensive. For example, computing the full graph automorphism group is as hard as solving graph isomorphism [152], for which no polynomial algorithm is known for the general case.*

However, our results are applicable even if we know only a subgroup of the full group of symmetries. Such subgroups may be known apriori or computed efficiently in some cases.

3.5.2.1 Application to the analysis of common problem classes

The Corollary 3.5.2.2 allows us to use graph theoretical apparatus to study the QAOA subspace of a given problem. We can now specify how this can be done for two popular problem formulations: QUBO and Ising forms.

Corollary 3.5.2.4 (QUBO and Ising subspaces)**.** *Consider the following quadratic unconstrained binary optimization problem (QUBO) on n variables:*

$$\min \sum_{1<i,j<n} Q_{ij} x_i x_j + \sum_{1<i<n} q_i s_i, \qquad x_k \in \{0,1\}$$

And an alternative formulation in Ising form:

$$\min \sum_{1<i,j<n} J_{ij} s_i s_j + \sum_{1<i<n} h_i s_i, \qquad s_k \in \{-1,+1\}$$

Note that the two formulations are equivalent up to additive constant. For a this QUBO (Ising) problem, consider a graph G on n nodes with the weight of node k equal to q_k (h_k) and the weight of the edge between nodes u, v equal to $Q_{u,v}$ ($J_{u,v}$). If $\sigma \in S_n$ is an automorphism of G preserving edge weights, $B = \sum_i X_i$ is the transverse field Hamiltonian and initial state is uniform superposition $|\psi\rangle = |+\rangle^n$, then QAOA is symmetric under $A(\sigma) \in S_{2^n}$.

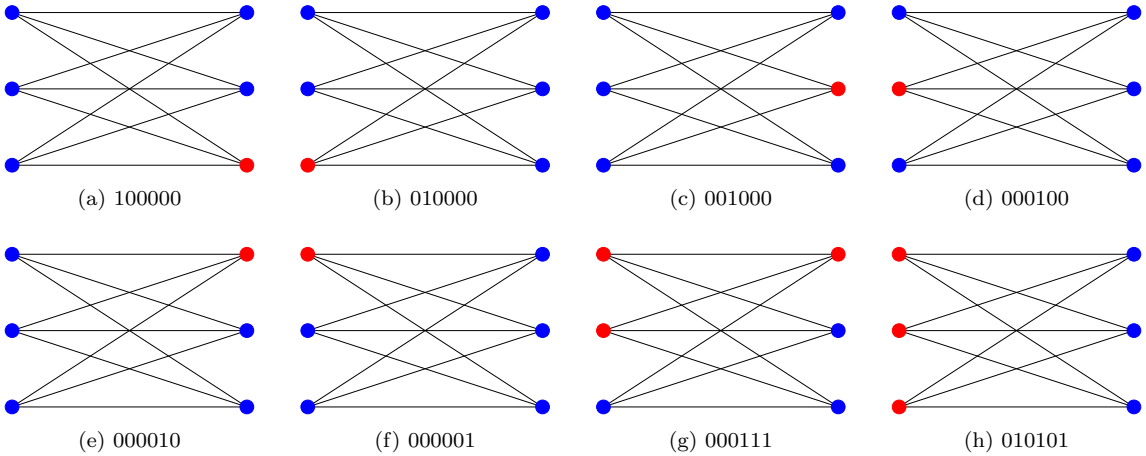
Proof. Trivial from Corollary 3.5.2.2. □

Remark. *Note that MaxCut problem we study in this Section can be formulated in Ising form by setting J equal to the graph adjacency matrix.*

This approach can be extended to higher-order problems by using hypergraphs. Hypergraph $H = (V, E)$ is a generalization of a graph where an edge is not limited to two nodes, but instead can connect an arbitrary subset of vertices.

Corollary 3.5.2.5 (Higher-order binary optimization subspace)**.** *Consider the following optimization problem on n variables:*

$$\min \sum_{e \in E} Q_e \prod_{v \in e} v_i, \qquad v_i \in \{0,1\}$$



(a) 100000 (b) 010000 (c) 001000 (d) 000100

(e) 000010 (f) 000001 (g) 000111 (h) 010101

Figure 3.11: All six red nodes in Figs. 3.11a to 3.11f are on the same vertex orbit (i.e., there exists an automorphism that takes one into another). Therefore from Corollary 3.5.2.2, the corresponding bitstrings will have the same probabilities in QAOA state. There is no automorphism that takes the red nodes in Fig. 3.11g into Fig. 3.11h.

Consider a hypergraph $H(Q)$ on n nodes with the weight of the edge connecting the nodes in subset e equal to Q_e. Here e is an edge containing variables corresponding to all terms in the product term $\prod_{v\in e} v_i$. Then if $\sigma \in S_n$ is an automorphism of $H(Q)$ preserving edge weights, $B = \sum_i X_i$ is the transverse field Hamiltonian and initial state is uniform superposition $|\psi\rangle = |+\rangle^n$, then QAOA is symmetric under $A(\sigma) \in S_{2^n}$.

Proof. Trivial from Corollary 3.5.2.2. □

3.5.2.2 Example: MaxCut on $K_{3,3}$ graph

To illustrate the results in the previous section, consider the example of the MaxCut on 6-node complete bipartite graph $K_{3,3}$. This graph has only one vertex orbit (see Fig. 3.11). Therefore from Corollary 3.5.2.2 all bitstrings with Hamming weight one will have the same probabilities in QAOA state $|\vec{\beta},\vec{\gamma}\rangle_p$, regardless of depth p or the choice of parameters $\vec{\beta},\vec{\gamma}$: $Pr_p(100000) = \ldots = Pr_p(000001)$. At the same time, there's no automorphism taking the part assignment represented by the bitstring 000111 into the part assignment represented by 010101 (see Figs. 3.11g and 3.11h). This does not mean that these two bitstrings will always have different probabilities. For example, for the trivial case of $p = 1, \beta = 0, \gamma = 0$, QAOA state reduces to uniform superposition over computational basis states: $|\vec{\beta},\vec{\gamma}\rangle_p = |+\rangle^n$. In this case $Pr_p(000111) = Pr_p(000111) = Pr_p(010101) = Pr_p(010101)$.

3.5.3 Using subspace symmetry to predict QAOA performance

In this Section, we apply the ideas introduced in Section 3.5.2 to study QAOA performance and its relationship with QAOA subspace structure. This requires quantifying two things. First, we have to specify a metric of "QAOA performance" for a problem instance. Second, we have to have a way of quantifying the structure of QAOA subspace. Note that from Corollary 3.5.2.2 both the structure of QAOA subspace as well as QAOA performance are in general specific to a particular problem instance (e.g., the MaxCut on unweighted $K_{3,3}$ graph) and not problem class (e.g. MaxCut on unweighted graphs) (See also Theorem 1 of [256].) Therefore we focus on studying the connection between properties of the particular instance and QAOA performance on that instance.

Let us begin with a metric of "QAOA performance". We discuss the choice of the QAOA parameters $\vec{\beta}, \vec{\gamma}$ in Section 3.5.3.1, but for the moment let us assume we have a classically efficient means of determining a sufficiently good set of parameters $\vec{\beta}, \vec{\gamma}$ for each given optimization problem instance. A central question for any approximate optimization algorithm is "can this algorithm achieve an approximation ratio of $(1-\epsilon)$ for a given instance?"[3] From the adiabatic limit, we know that as $p \to \infty$, QAOA can solve any optimization problem exactly [75]. Therefore, the answer to that question is "yes, with sufficiently high p". However, the running time of QAOA grows linearly with its depth p, so p cannot grow faster than polynomially with problem size for such approaches to remain efficient. Moreover, in the NISQ era the depth p is further limited by the error rate of the hardware. Therefore a more practical question to ask is "what is the smallest p with which QAOA achieves an approximation ratio of $(1-\epsilon)$ for a given instance?" This number ($p_{\min} = p_{\min}(C, B, |\psi\rangle)$) is a meaningful measure of QAOA performance: the smaller $p_{\min}$ is, the lower is the time-to-solution for QAOA[4]. Now all we need to do to define a metric of QAOA performance is a reasonable way of finding parameters (schedules) $\vec{\beta}, \vec{\gamma}$, which we present in Section 3.5.3.1.

The second part of the proposed approach to analyzing QAOA is a metric or an index of QAOA subspace symmetry. The connection in Corollary 3.5.2.2 allows us to study QAOA subspace for a particular instance by studying the symmetry of the corresponding graph representation. For MaxCut problem, this duality is trivial, as graph representation of the optimization problem is exactly the graph on which we want to solve MaxCut. We discuss the symmetry metrics we use to

[3]Note that here we approach QAOA as a *heuristic* applied to a particular problem *instance*. In doing this, we are explicitly *not* making a general statement about QAOA solving classes of problems efficiently to a given approximation ratio. Rather, we are interested in defining a reasonable measure of QAOA performance on a particular instance.

[4]We are explicitly not making any predictions about whether $p_{\min}$ grows exponentially with problem size in worst case. Rather, we use it as a metric to understand QAOA performance on specific problem instances.

measure the symmetry in Section 3.5.3.2. We study both exact and approximate symmetry.

Finally, in Section 3.5.3.3 we treat the metrics of the graph symmetry presented in Section 3.5.3.2 as *instance features*. We observe that they correlate with the chosen measure of QAOA performance (namely, $p_{\min}$), and use them as input to machine learning models that we train to predict $p_{\min}$. The success of these models, which only have access to the information about graph symmetry, is indicative of the power of the proposed approach.

3.5.3.1 Smooth schedules: a practical approach to QAOA

One challenge in both the analysis and implementation of QAOA is the need to sufficiently optimize the algorithm parameters $\vec{\beta}, \vec{\gamma}$. It is known that optimizing parameters for QAOA, e.g., variationally, appears to be difficult in practice [224, 220, 133], which in particular presents two challenges for numerical experiments. First, performing such numerical studies requires solving the parameter optimization problem for a large number of instances, which is difficult in practice due to the lack of fast high-quality optimization methods. Second, even if such data were collected, it would likely not be representative of the QAOA performance, as finding optimal parameters for every instance is not a practical way of using QAOA as the depth p becomes large.

Instead, there are two practical approaches to using QAOA. The first approach, explored in [133, 259, 250], is choosing a class of instances and training a machine learning model to quickly produce high quality QAOA parameters for this class of instances. While powerful, this approach only resolves the second challenge, as to train the model one still has to find optimal parameters for a sufficiently large dataset representative of the chosen class of instances. The second approach, explored in this Section, is inspired by the smooth schedules observed in Ref. [267, 59], and in particular by the reparameterization developed in [267]. In this approach, we specify a class of *parameter schedules*, and then fit a schedule of this class to a particular problem instance. The goal of this restriction is to dramatically reduce the difficulty of the parameter optimization problem. There are various reasonable choices for the class of schedules; in this work, we focus on *smooth* schedules, i.e., schedules where the change from one component to the next in $\vec{\beta} \in \mathbb{R}^p$ and $\vec{\gamma} \in \mathbb{R}^p$ is small relative to the difference between

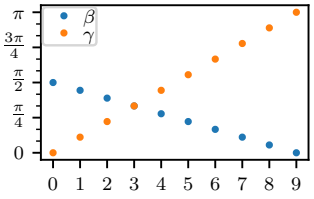


Figure 3.12: Example of a linear schedule for $p = 10$. β_j changes linearly from $\pi/2$ at the first QAOA step to 0 at the last QAOA step, and γ_j changes from 0 to π.

minimum and maximum values of β and γ. Specifically, we focus on *linear schedules*, i.e., schedules where the components $\beta_j, \gamma_j, 1 \leq j \leq p$ change linearly with step j (an example of such schedule is presented in Fig. 3.12).

Restricting QAOA parameters to a particular class of schedules clearly yields a tradeoff between the difficulty of parameter optimization and the algorithm's performance, relative to unrestricted parameters. Fitting a particular class of schedules is much easier than finding an optimal general schedule. For example, in general for $p = 10$ finding optimal parameters requires optimizing $2 \times p = 20$ parameters, whereas fitting linear schedule only requires to optimize over 4 parameters (slope and intercept for both β and γ). We also observe this in practice, with multistart methods [224] capable of finding optimal linear schedules quickly and reliably. At the same time, this restriction may increase the minimum depth needed to achieve the desired approximation ratio (i.e., $p_{\min}(C, B, |\psi\rangle) > p_{\min}(C, B, |\psi\rangle \text{, linear schedule})$). Note that in adiabatic limit $p \to \infty$ linear schedules are sufficient, so such minimum depth $p_{\min}$ always exists. For the rest of this Section we will only consider QAOA with linear schedules: $p_{\min} = p_{\min}(C, B, |\psi\rangle \text{, linear schedule})$.

3.5.3.2 Measuring subspace symmetry

Quantitative analysis of graph symmetry is a challenging problem. Over the years, multiple symmetry indices have been introduced to attempt to tackle this problem. Some notable examples are graph entropy [162, 232], index of symmetry [163], network redundancy [148] and normalized network redundancy [19]. However, we observe that all of those metrics fail to capture fine difference between instances, required for achieving high predictive power. Therefore we propose constructing a data-driven symmetry index that is tailored to our particular application. We construct this index by noting that all of the metrics outlined above are a combination of the number of vertex orbits, the size of orbits, the size of group of automorphisms, the number of nodes of a graph and the graph entropy. Instead of combining them in an index constructed from some prior intuition above graphs, we use them as features in a machine learning model. The resulting trained model serves as a symmetry measure of a graph.

First, use the following *exact* symmetry measures:

1. Logarithm of the size of the group of automorphisms of the graph: $\log |Aut(G)|$
2. Number of vertex orbits of the graph: $|O(G)|$

3. Entropy of the graph: $I(G) = \frac{1}{n}\sum_i |A_i| \log |A_i|$, where $|A_i|$ is the size of i-th vertex orbit and n is the number of nodes in G.

Second, we add the following *approximate* symmetry measures to further improve the performance of our model. Let G'_e denote a graph constructed from G by removing edge e. Intuitively, if G'_e has an automorphism σ, then σ is an *approximate* automorphism of G. Therefore for each measure of exact symmetry of G, we can introduce a measure of *approximate* symmetry of G defined as the average value of such measure on G'_e, averaged across all edges e. Similarly, we consider G''_{e_1,e_2} that is constructed by removing two edges (e_1, e_2) from G, and use the symmetry of G''_{e_1,e_2} averaged across all pair of edges e_1, e_2 as an measure of approximate symmetry of G. This adds another six metrics to the list of features:

1. Average size of group of automorphisms for graphs constructed from G by removing one edge: $\frac{1}{h'}\sum_{i=1}^{h'} \log |Aut(G'_i)|$, where $h' = \binom{|E|}{1} = |E|$
2. Average size of group of automorphisms for graphs constructed from G by removing one edge: $\frac{1}{h''}\sum_{i,j} \log |Aut(G''_{i,j})|$, where $h'' = \binom{|E|}{2}$
3. Average number of vertex orbits for graphs constructed from G by removing one edge: $\frac{1}{h'}\sum_{i=1}^{h'} |O(G_i)|$, where $h' = \binom{|E|}{1} = |E|$
4. Average number of vertex orbits for graphs constructed from G by removing two edges: $\frac{1}{h''}\sum_{i,j} |O(G''_{i,j})|$, where $h'' = \binom{|E|}{2}$
5. Average entropy of graphs constructed from G by removing one edge: $\frac{1}{h'}\sum_{i=1}^{h'} I(G_i)$, where $h' = \binom{|E|}{1} = |E|$
6. Average number of vertex orbits for graphs constructed from G by removing two edges: $\frac{1}{h''}\sum_{i,j} I(G''_{i,j})$, where $h'' = \binom{|E|}{2}$

Finally, we add the number of nodes in the graph as a feature, for a total of ten features.

3.5.3.3 Learning the relationship between symmetry and performance

To numerically study the relationship between QAOA subspace symmetry and QAOA performance we use the MaxCut problem on a dataset of 294 graphs, selected as described in Table 3.1 below. We set the tolerance in the definition of $p_{\min}$ to be 0.05, i.e., $p_{\min}$ ="the smallest p with which

QAOA achieves an approximation ratio of 0.95 for a given problem." The choice of the tolerance is not special in any way, as long as there is not efficient classical algorithm that can achieve that approximation ratio in the worst case. In the case of MaxCut, the best approximation ratio in the worst case is 0.87856 under the Unique Games conjecture, achieved by the Goemans-Williamson algorithm [90]. Without the assumption of Unique Games conjecture, it is known that for MaxCut it is NP-hard to achieve approximation ratio better than $\frac{16}{16} \approx 0.941$. Both of those approximation ratios are lower than our chosen target approximation ratio of 0.95.

An important caveat is that while the choice of 0.95 approximation ratio does appear to imply that the problems we are solving are "classically hard", it is not necessarily the case. The bounds we mention above are *worst case*, and identifying hard instances for which the worst-case conditions are triggered is hard. For many of the graphs in our benchmark (e.g. the cycle graph) the MaxCut problem is trivial classically. Moreover, all the problems we consider are bounded in size (the largest one has only 22 nodes), making it impossible to make any kind of asymptotic argument. Rather, the correct way to understand the choice of approximation ratio of 0.95 is by viewing at it as "classically non-trivial" regime. In other words, choosing approximation ratio above the ones that are known to be classically hard opens a possibility of quantum advantage, whereas for approximation ratios known to be efficiently attainable classically, no such possibility exists.

Name	# graphs	min $\|V\|$	max $\|V\|$	min $\|E\|$	max $\|E\|$
Antiprism	7	6	22	12	44
Circular Ladder	7	6	18	9	27
Complete	18	3	20	3	190
Cycle	18	3	20	3	20
2D Grid	4	16	21	24	42
Hand-picked	7	10	20	15	30
Ladder	7	6	18	7	25
Random 3-regular	60	10	20	15	30
Random 4-regular	59	10	20	20	40
Random 5-regular	60	10	20	25	50
Star	18	4	21	3	20
Trivial	13	7	19	6	18
Wheel	16	5	20	8	38
Total	294	3	22	3	190

Table 3.1: Description of the dataset. "Trivial" is graph with a trivial group of automorphisms. "Hand-picked" includes various textbook graphs with large groups of automorphisms, e.g. Peterson and Heawood graphs. We make the full dataset available online [5]

We use nauty [158] to compute the features. We compute $p_{\min}$ for each problem in the

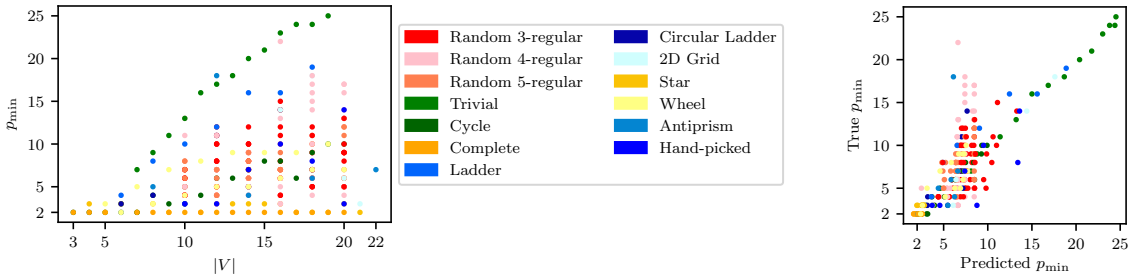
dataset by considering iteratively larger depths p, starting with $p = 2$, and optimizing the linear schedule for this depth, until we achieve the desired approximation ratio. We use COBYLA [194, 196] implemented in SciPy [122] package as a local optimizer in the libEnsemble [116] implementation of APOSMM [138, 139]. We use NetworkX [101] for graph operations and GNU-Parallel for large-scale experiments [240]. The code is available online at [1].

Initial observations First, we observe correlation between the chosen features and $p_{\min}$. Correlation coefficients are presented in Table 3.2, where Pearson correlation coefficient is defined as $\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{x})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}}$, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the sample mean. Note that measures of approximate symmetry (e.g. average size of group of automorphisms for G') have similar correlation coefficients to the corresponding exact symmetries, as expected.

Feature	Pearson r	p-value
$\log Aut(G)$	-0.36397	0.00000
$\frac{1}{h'}\sum_{i=1}^{h'} \log Aut(G'_i)$	-0.32851	0.00000
$\frac{1}{h''}\sum_{i,j} \log Aut(G''_{i,j})$	-0.30854	0.00000
$\lvert V\rvert$	+0.37075	0.00000
$\lvert O(G)\rvert$	+0.39314	0.00000
$\frac{1}{h'}\sum_{i=1}^{h'} \lvert O(G_i)\rvert$	+0.44326	0.00000
$\frac{1}{h''}\sum_{i,j} \lvert O(G''_{i,j})\rvert$	+0.43035	0.00000
$I(G)$	-0.32295	0.00000
$\frac{1}{h'}\sum_{i=1}^{h'} I(G_i)$	-0.36913	0.00000
$\frac{1}{h''}\sum_{i,j} I(G''_{i,j})$	-0.34254	0.00000

Table 3.2: Pearson correlation coefficient r and p-value for the test of non-correlation of features with $p_{\min}$. See Section 3.5.3.2 for the definitions of the features.

Second, we observe that the hardest for QAOA problems in our dataset are the problems corresponding to the least symmetric graphs and the easiest are the ones corresponding to the most symmetric graphs. Fig. 3.13a shows the scaling of $p_{\min}$ with problem size. The hardest problems are the ones for which $p_{\min}$ grows quickly with problem size (i.e., QAOA depth required to achieve the desired approximation ratio grows quickly with problem size) and the easiest problems are the ones for which $p_{\min}$ grows slowly or does not grow at all. Fig. 3.13a shows that for problems with no symmetry (graphs with trivial group of automorphisms of size 1), $p_{\min}$ grows the fastest, and for problems with the most symmetry (all permutations of nodes are an automorphism for complete graph) $p_{\min}$ grows the slowest. This supports the intuition that there exists a connection between symmetry and QAOA performance.



(a) $p_{\min}$ as a function of the number of nodes. $p_{\min}$ grows the fastest for the least symmetric graphs ("Trivial") and the slowest for the most symmetric (complete and star graphs).

(b) True $p_{\min}$ and $p_{\min}$ predicted by Support Vector Regression for both training and testing data. Median absolute error on the test set is 1.37.

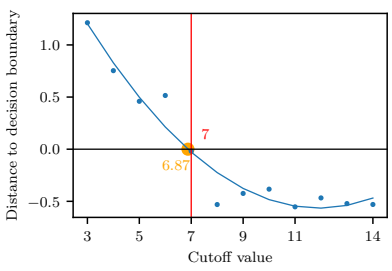
Figure 3.13: Using problem symmetry to predict $p_{\min}$.

Machine learning approaches to predicting QAOA performance To quantify the intuition arising from Fig. 3.13a, we use the metrics discussed in Section 3.5.3.2 as features for a machine learning model that we train to predict $p_{\min}$. We reserve 30% on the dataset as the testing set and use the rest for training the model. We approach the problem in two ways. In the first approach, we treat the task of predicting $p_{\min}$ as a regression problem. In the second approach, we use an ensemble of classifiers to predict $p_{\min}$. Below we discuss both approaches and the trade-offs they introduce. We choose median absolute error as our target metric. This choice is motivated by the physical meaning of $p_{\min}$, namely, the minimum depth required to achieve 0.95 approximation ratio by using QAOA with linear schedules. As $p_{\min}$ is an integer, absolute error is an easily interpretable metric. Moreover, absolute error of one or close to one is tolerable for most applications (e.g. if we want to use the trained model to predict the value of depth p to use in QAOA or if we want to establish whether a given problem requires depth beyond the capabilities of target quantum hardware).

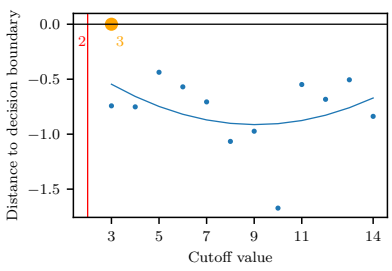
First, we approach the problem of predicting QAOA performance (i.e., predicting the number $p_{\min}$) directly as a regression problem. We use Support Vector Regression (SVR) [69, 189, 51] with radial basis function kernel. We use 5-fold cross-validation stratified by graph class for hyperparameter optimization. We achieve 0.73 median absolute error on training set and 1.37 on the testing set (i.e., on the instances previously unseen by the model). We observe correlation between predicted $p_{\min}$ and true $p_{\min}$ with Pearson correlation coefficient of 0.71 on the test set. The results are visualized on Fig. 3.13b.

Second approach we use for predicting $p_{\min}$ is training an ensemble of classifiers. We simulate

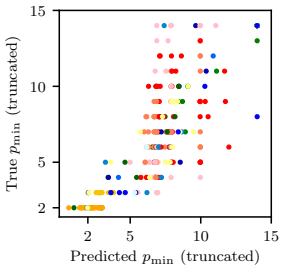
a realistic scenario of limited circuit depth by grouping all instances with $p_{\min} \geq 15$ into one class corresponding to "depth beyond the capabilities of the target hardware". As $p_{\min}$ is a discrete value (an integer), classification is a natural choice. An issue with using a classifier directly is that assigning each value of $p_{\min}$ to be a class discards the information that two consecutive integers are more similar than two integers that are far apart. To address this, we instead train an ensemble of binary classifiers, where each classifier answers the question "is the value of $p_{\min}$ smaller than specified cutoff?" Using an ensemble of "cutoff" classifiers is a standard approach for ordinal regression [253, 191]. Each classifier is a Support Vector Machine classifier (SVC) [189, 51]. We use the radial basis function kernel and the optimal hyperparameters found by cross-validation for the Support Vector Regression.



(a) Blue dots are distance to decision boundary for each classifier. Blue line is the least squares quadratic fit. Yellow dot is the predicted $p_{\min}$ and the red vertical line is the true $p_{\min}$.



(b) If the least squares quadratic fit does not intersect with x-axis within the bounds, we choose either minimum of maxium $p_{\min}$ based on the majority vote of classifiers. Here all classifiers put the point in first class ("$p_{\min}$ is smaller than the cutoff").



(c) True $p_{\min}$ and $p_{\min}$ predicted by an ensemble of Support Vector Machine for both training and testing data. Median absolute error on the test set is 1.35. See Fig. 3.13 for the color legend.

Figure 3.14: Using an ensemble of SVM classifiers to predict $p_{\min}$.

The ensemble of "cutoff" binary classifiers is used to predict $p_{\min}$ in the following way. For a new point, we compute the distance to the decision boundary for each classifier. By convention, if the distance is positive, then the point belongs to the first class ($p_{\min}$ is smaller than the cutoff), else it belongs to the second class ($p_{\min}$ is greater or equal to the cutoff). Therefore minimum cutoff value at which the binary classifiers are assigning the point to the second class (i.e., at which the distance to the decision boundary changes sign) is the $p_{\min}$ for this point. To make the distances to the decision boundary directly comparable for all classifiers, we standardize them in the following way. For each classifier we compute distances to the decision boundary on the entire training set. If the standard deviation of these distances is σ, we let the standardized distance to decision boundary

be $d_z = d/\sigma$, where d is the distance to the decision boundary for the new point. To predict $p_{\min}$ for the point, we fit a least squares quadratic function to the standardized distances to the decision boundary as a function of the cutoff value for all classifiers, and use its intersection with x-axis as the predicted value. This process is visualized in Figs. 3.14a and 3.14b. We achieve median absolute error of 1.35 on the test set.

Both approaches demonstrate very similar performance. We observe high ratio of support vectors, indicating that the performance can be further improved by introducing more datapoints (problem instances).

3.5.4 Discussion

In this Section we introduce a novel connection between QAOA subspace symmetry and the symmetry in the (hyper)graph representation of the problem. We show the power of this connection by training machine learning models to predict QAOA performance on a given MaxCut instance solely from the information about exact and approximate symmetries in the graph.

A limitation to applying our machine-learning approach in practice is that many of the metrics defined in Section 3.5.3.2 cannot be computed in polynomial time in general. While for small problem instances off-the-shelf tools like nauty [158] can compute them in seconds, this is not a scalable approach. Therefore a crucial next step is coming up with ways to compute a heuristic approximate metric of symmetry, both exact symmetry as well as approximate symmetry. Techniques like network alignment [197] can be applied to quickly measure the amount of symmetry in the graph. Interpretability of the machine learning approach may also evolve with both the size of instances and consequently the training set size. Although, the SVM/SVR approach is one of the most interpretable learning models that helps to estimate the importance of features and their combinations, advanced scalable nonlinear methods such as [210] will be required to avoid over-fitting of the model.

Second important future direction is replacing global symmetries with local ones. As QAOA is a local algorithm in constant depth regime [78], local symmetries (e.g. two small subgraphs of G as isomorphic) should be more a more precise predictor of QAOA performance than global symmetries. We take a step in this direction by considering approximate symmetries, which can be understood as "almost" global: a permutation is an approximate automorphism of G if it is an isomorphism of two subgraphs of G. Further research in this direction is complicated by the fact that local symmetries

only come into play for larger graphs, limiting our ability to perform numerical studies.

Chapter 4

Decomposition-based Hybrid Quantum-Classical Algorithms

The number of qubits in NISQ-era devices available at the time of writing is not nearly enough to demonstrate quantum advantage, which makes it especially hard to demonstrate the usefulness of quantum computers to solve real problems. For example, the possibility of quantum speedup using the hybrid quantum approximate optimization algorithm (QAOA) for a network problem similar to the one discussed in this Section (max-cut) is a subject of active discussion. On one hand, there are theoretical results demonstrating that QAOA for max-cut problem improves upon best know classical approximation algorithms for certain graphs [186, 256]. At the same time, there are indications that achieving speedup using QAOA might require at least several hundred qubits [95]. Research and development of quantum algorithms is necessary as the number and quality of qubits is improving. These quantum algorithms can also be used to improve classical algorithms [239]. The need for development of new quantum algorithms was highlighted in the recent National Academy of Science report [94]. One of the important directions to make quantum computing feasible in the near future is to use various problem decomposition approaches to solve a large problems as a set of subproblems. This can be accomplished at various levels such as problem formulation or at the algorithmic level as demonstrated in this Section.

The decomposition approach might be the key method to achieve a quantum speedup on even modest-size NISQ devices in near-term future. To support this claim, there is an important

and encouraging work [30], where it was shown that large combinatorial optimization problems can be effectively decomposed into subproblems on quantum annealing hardware, while still obtaining high quality of the overall solution. It was demonstrated for solving embedding problems on D-Wave quantum computers, but we believe that the same technique can be used to improve dramatically the speed and performance of QAOA algorithms on universal quantum computers.

4.1 Quantum Local Search for Network Community Detection on Small Quantum Computers

In this Section, we introduce the quantum local search (QLS) algorithm for the network community detection problem that is based on the local search method [6]. Many different versions of the local search have been applied to numerous computationally hard problems such as the satisfiability testing [219], and the traveling salesman problem [261, 120]. Local search is used for problems where a global solution cannot be computed directly but instead can be iteratively approximated in the space of candidate solutions (sub-problems), until optimal (or sufficiently good). The important feature of QLS is that it is a hybrid hardware-agnostic algorithm that combines a classical machine with a small quantum device. In this method, QLS allows us to leverage available NISQ-era quantum devices to solve machine learning problems of practical size for the first time.

A version of the network community detection (also known as graph clustering) is an unsupervised machine learning problem used to identify sub-structure as communities in such networks as computer and information infrastructures, social activities, and biological interactions or co-occurrences. It is used to find non-trivial topological features, with patterns of connection between nodes that are neither exactly regular nor random. For example, in metabolic networks, communities correspond to a series of chemical reactions called metabolic pathways [105], whereas in a protein interaction network, communities correspond to proteins with similar functionality inside a biological cell [53]. In this work we focus on using Newman's modularity-based community detection [176].

QLS was applied to solving the 2-community detection problem on real networks of up to 410 nodes, while solving a 16 variable subproblem on a quantum device. To the best of our knowledge, this is the first attempt to tackle problems of this size using gate-model (universal) quantum computing. Also, QLS is shown to work with the D-Wave quantum annealer. We explore the potential of QLS as quantum devices become more and more capable and demonstrate its

potential.

The small size of available quantum devices creates a challenge, since typical algorithms (both quantum and classical) look at a problem "as a whole", requiring large amounts of resources to store the description of the entire problem. While on classical computers storing the problem usually does not constitute a problem, it becomes a bottleneck when working with quantum computers that only have limited numbers of qubits and limited connectivity between qubits. The number of variables that can be represented in a quantum device is dependent on its underlying architecture.

A problem decomposition approach like local search presents a natural solution to this problem. A local search heuristic starts with some initial solution and searches its neighborhood iteratively, trying to find a better candidate solution with improved criterion (which is often an objective of the corresponding minimization or maximization of the problem). If a better solution is found, it replaces the current solution, and the search continues [6]. Searching the neighborhood is a local problem and its size can be restricted to fit on a small quantum device. In QLS for graph community detection, the neighborhood of the solution is searched by selecting a subset of vertices and collectively moving them between the communities with the goal of improving the global modularity metric.

The QLS approach provides an additional benefit of being fundamentally hardware-agnostic. Local neighborhood search can be encapsulated as a routine, allowing researchers to easily switch between different hardware implementations. This is especially useful, since the landscape of quantum computing in the NISQ era is in a constant state of flux with many QC architectures available and new development happening constantly. It is not clear at this stage which architecture will become dominant in future. In this work we demonstrate how the two most developed and popular current paradigms, universal quantum computing (UQC) and quantum annealing (QA), can be integrated into the QLS framework and utilized to solve problems of practical size. Both paradigms have demonstrated great potential on a number of important problems [134, 202, 14, 70].

In this Section, we do not aim to analyze performance of quantum optimization algorithms like quantum annealing or QAOA. Although we do present some performance results (see Fig. 4.2), they by no means constitute an exhaustive comparison with classical state-of-the-art. Instead, they provide motivation for our work, demonstrating that the subproblems offloaded to quantum solvers are not trivial and that hybridization is needed. For benchmarking, analysis and exploration the reader is referred to one of a number of recent papers analyzing QAOA performance [267, 95, 186].

In other words, we do not focus on finding and quantifying quantum speedups. Instead, we focus on a different question: if these algorithms are indeed capable of providing speedups in the near term, how can we leverage them to solve practical problems?

It is important to point out that the introduction of a problem decomposition heuristic like QLS limits the possible quantum speedup. Since to the best of our knowledge no asymptotic speedups have been shown so far for QAOA or QA, decomposition schemes limit the multiplicative speedup on the entire global problem by the multiplicative speedup on a small local subproblem. However, they still provide a way to take advantage of the small quantum devices that are becoming available.

4.1.1 The Community Detection Problem

For an introduction to the community detection problem the reader is refered to Section 3.4.1.

Community detection using a hybrid quantum-classical approach targeted for specific quantum architectures has been demonstrated previously. The 2-community problem was solved using *qbsolv* and the D-Wave quantum annealer [245] and extended for k-communities [161, 172]. Solving for 2-communities using QAOA and the IBM Q Experience was shown in [161]. Solving for k-communities on signed graphs using block coordinate descent [269, 206] and D-Wave quantum annealer was shown in [265].

4.1.2 Quantum-accelerated Decomposition Heuristics for Optimization

Central to the discipline of QC in the NISQ era is the problem of a limited number of available noisy qubits. For example, at the time of writing, the largest gate-model QC device available on the cloud was IBM Q 20 Tokyo [58] with twenty superconducting qubits. Twenty qubits translates into up to 20 variables due to connectivity constraints. This implies that the maximum number of nodes of a network we can cluster directly is 20. This example highlights the challenges of leveraging limited NISQ-era devices to solve practical problems and motivates our local-search approach. Note that same considerations apply for problems other than optimization. For example, similar hybrid approaches have been applied to Blind Quantum Computation [88, 25, 227, 229], and distributed quantum machine learning [228]. Parallel Quantum Computation (PQC) [145] can

be used to speed up Grover's search algorithm [93] by dividing a database on which the search is performed between an ensemble of quantum computers running in parallel [262, 144].

In response to the challenges of quantum computation in the NISQ era, a number of decomposition approaches have been explored. The methods described in this section use limited in size quantum optimization solvers to search a restricted neighborhood of a given solution with the goal of finding a better solution. Here the given solution comes either from running a classical heuristic solver on a CPU or from the previous iteration. These methods are inspired by the success of classical large-scale neighborhood local search methods (the reader is referred to [207] for a survey of local-search heuristics in general and to [8] for a survey of large-scale neighborhood methods in particular). It is important to note that unlike this Section, all the works described in this section focus exclusively on D-Wave quantum annealers.

The first family of methods builds on classical pre-processing methods for quadratic unconstrained binary optimization (QUBO) problems (see [241] for a review). One such pre-processing technique is heuristically fixing variables. The variables are chosen by maintaining a set of elite solutions and fixing the variables that have the same value across many or all local optima, with the intuition being that they will have the same values for the global optimum [255]. Sample persistence variable reduction (SPVAR) [125] in its basic version uses a sample of solutions (obtained either from a quantum annealer or a classical heuristic) and fixes the variables that have the same value across the entire sample. Then SPVAR uses a quantum annealer as the solver for the restricted QUBO. This method was later extended by introducing multistart (multiple samples) and was extensively benchmarked using both the D-Wave quantum annealer as well as state-of-the-art classical heuristics for Chimera Hamiltonians [126].

The second family of methods extends iterative large-scale neighborhood local search methods. Local search commonly considers the neighborhood of bit strings that have Hamming distance one from the current solution at each step. The performance of local search methods can be improved by considering larger neighborhoods (Liu et al. [142] shows significant performance improvements for neighborhood of Hamming distance four, equivalent to fixing all but four variables). Quantum optimizers provide a potentially efficient way to explore these larger neighborhoods. This rather straightforward idea was introduced in [29] and extended and rigorously tested in [30, 206, 269]. A similar hybrid tree search method was presented in [243]. These methods utilize the D-Wave quantum annealer as the quantum optimizer, enabling them to solve problems with thousands of

variables. In this work we limit the subproblem size to be small enough to fit on the IBM Q quantum computer, limiting the size of the problems we can tackle. D-Wave provides a set of utilities for problem decomposition, including a hybrid extension of the tabu search QSage [63].

4.1.3 Quantum Local Search

To address the challenges outlined above, we introduce the QLS algorithm. QLS is a hybrid quantum-classical local-search approach, inspired by numerous existing local-search heuristics. QLS is motivated by the successful application of local-search heuristics to a variety of optimization problems. The novelty of QLS is that it can utilize both quantum annealers and universal quantum computers. In this work, we apply QLS to the problem of 2-community detection on graphs, but the success and versatility of local-search heuristics make us confident that QLS can be extended to other optimization problems.

In QLS for community detection, the local search starts with a random assignment of communities to vertices and attempts to iteratively optimize the current community assignment of a *subset* of vertices with the goal of increasing modularity. Here the space of potential community assignments of a subset of vertices plays the role of the neighborhood where the local search is performed. At each iteration, a subset $X \subset V$ is populated by selecting vertices with the highest potential gain in modularity obtained when changing their community assignment. This can be done efficiently [176] since at each iteration we only need to update the gains of vertices in X and their neighbors. Then at each iteration, the community assignment of the vertices in the subset X (subproblem) is optimized using a routine that includes a call to a quantum device. The local search proceeds until it converges. We define convergence as three iterations with no improvement in modularity. Note that in general it is not necessary to consider all vertices before convergence: in the 2-community problem, random initial assignment would be correct for 50% of vertices on average. Our approach is outlined in Algorithm 1.

Algorithm 1 QLS Community Detection

solution = initial_guess(G)

while *not converged* **do**

  X = populate_subset(G)

  // using IBM UQC or D-Wave QA

  candidate = solve_subproblem(G, X)

  if *candidate* > *solution* **then**

    solution = candidate

  end

end

The subproblem of optimizing community assignment of the subset is formulated by fixing community assignment for all vertices not in the subset ($i \notin X$) and encoding them into the optimization problem as boundary conditions. This is a commonly used technique in many heuristics [141, 103]. Denoting fixed assignments by $\tilde{s}_j$, the subproblem can be formulated as:

$$\begin{aligned} Q_s &= \textstyle\sum_{i>j|i,j\in X} 2B_{ij}s_is_j + \sum_{i\in X}\sum_{j\notin X} 2B_{ij}s_i\tilde{s}_j \\ &= \textstyle\sum_{i>j|i,j\in X} 2B_{ij}s_is_j + \sum_{i\in X} C_is_i, \\ \text{where } C_i &= \textstyle\sum_{j\notin X} 2B_{ij}\tilde{s}_j \end{aligned} \tag{4.1}$$

Clearly, maximizing (4.1) can only increase global modularity (3.7). The objective defined in Eq. (4.1) can be optimized using a QC algorithm. The exact way the optimization is performed can vary between different QC implementations, making our approach extendable to new emerging QC platforms. We demonstrate this portability by implementing two subproblem optimizing routines that use IBM Q 16 Rueschlikon [58] and D-Wave 2000Q [61]. Additionally, we implement a subset optimization routine that uses the classical Gurobi solver [181] for quality comparison. The choice of Gurobi is not of importance, since for subproblems with 16 variables any classical integer programming solver is capable of finding the optimum.

4.1.4 Results and discussion

We implement the classical part of QLS in Python 3.6, using NetworkX [101] for network operations. The subproblem solvers are implemented using QA (D-Wave SAPI), QAOA (IBM QISKit [60]) and the classical Gurobi solver [181]. Our framework is modular and easily extendable,

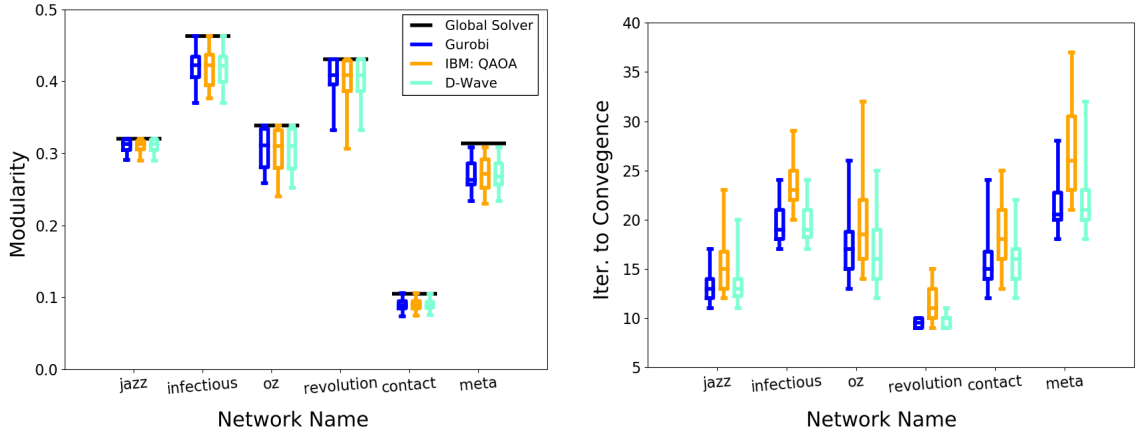


Figure 4.1: Box-plots showing the range of modularity scores for 2-community detection (left, greater is better) and number of solver calls (right, less is better) respectively for the three different subproblem solvers. The results show that the proposed approach is capable of achieving results close to the state-or-the-art (Global Solver)

allowing researchers to add new subproblem solvers as they become available. The framework is available on GitHub at `http://bit.ly/QLSCommunity`.

In order for a subproblem to be solved on the D-Wave system, the problem is embedded onto the physical layout (Chimera graph). The clique embedder [35] is used to calculate an embedding of a complete 16-variable problem once and is reused for each subproblem. In this work, we utilized D-Wave's Solver API (SAPI) which is implemented in Python 2.7, to interact with the system. We used the D-Wave 2000Q which has up to 2048 available qubits. Subproblems of approximately 64 variables can be solved on the the 2000Q, however, for a fair comparison, we limit ourselves to up to 16 variables. The D-Wave system is intrinsically a stochastic system, where solutions are sampled from a distribution corresponding to the lowest energy state. For each subproblem, the best solution out of 10,000 samples is returned.

The QAOA subproblem solver is implemented using the IBM QISKit framework. We ran QAOA with RYRZ ansatz [57] on the IBM 16 Q Rueschlikon [58] with 16 qubits. For optimization of the variational parameters we used a SciPy [122] implementation of Constrained Optimization BY Linear Approximation (COBYLA) method [194]. For each subproblem, we performed optimization of the variational parameters θ using a high-performance simulator [260] and ran QAOA with optimized parameters on a quantum device using the IBM Q Experience [58] cloud service. We allowed COBYLA 100 function evaluations (i.e. 100 QAOA runs on the simulator) to find optimal

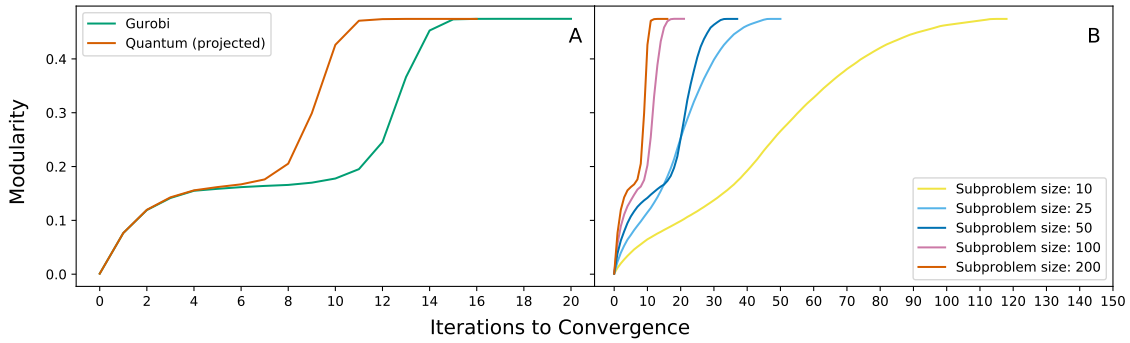


Figure 4.2: A projection of QLS performance as the hardware size of quantum devices increases. **(A)** Projected of QLS performance as the quality of the local search solver solution is improved. The projection is performed by comparing the performance of classical solver Gurobi with time limit fixed at 0.25s (D-Wave time to solution) and Gurobi with time limit 1000s (projected good solution). The assumption is that the new quantum optimization algorithms would be able to scale and provide results of the same quality as Gurobi with time limit 1000s while taking approximately the same time to solve the problem as they do today. **(B)** Projected number of iterations for QLS to converge as larger devices become available (projection performed by using Gurobi as a subproblem solver).

parameters θ. We used this setup (training on a simulator and running on the quantum device) because of the limitations of the IBM Q Experience job queue at this time. In our experience, jobs submitted to the IBM quantum device can spend minutes to hours in queue, requiring days to complete a full variational parameter optimization loop. It is our understanding that this will be remedied in the future. The main downside of this setup is that the variational parameters trained on a simulator do not encode the noise profile of the device, decreasing the quality of the solution. This is one of the main factors contributing to slightly slower convergence for QAOA compared to other methods. In the future, as various QC devices become available, it will be straightforward to perform QAOA fully on a QC device. However, even using the current setup we achieved very promising results, indicating great potential for applying variational quantum-classical methods to combinatorial optimization problems.

Our results are presented in Fig. 4.1. We ran our algorithm on six real-world networks from the KONECT dataset [136] with up to 410 nodes as our benchmark. The networks come from different real world phenomena and include social and metabolic networks. For each network, we ran 30 experiments with different random seeds. The same set of seeds was used by the three subproblem solvers, with all solvers starting with the same initial guess and therefore making the results directly comparable. We fixed the subproblem size at 16 vertices. Our results demonstrate that QLS with

both D-Wave QA and QAOA on IBM Q as quantum subproblem solvers perform similarly in terms of quality of the solution (modularity) and the number of iterations to convergence, and are capable of achieving results comparable to state-of-the-art. Our results are compared to results using the Gurobi Optimizer, which is a state-of-the-art solver for mathematical programming. We use the Gurobi Optimizer in two ways: first as a solver for solving the entire problem at once, which we report as the *Global Solver* and second as a solver for solving small size subproblems of fixed size within the local search framework. For solving the entire problem, the Gurobi Optimizer is unable to reach a provable global optima for most of the problems within the specified time-frame. For the graph problems of up to approximately 400 variables, we run Gurobi (as a global solver) for up to 72 hours and the results reported are within an optimality gap of up to 33%. For the smaller size subproblems of 16 variables, Gurobi was able to find the optimal solution within less than second. The networks and the set of seeds we used are available online at `http://bit.ly/QLSdata`.

The results demonstrate the promise of the proposed approach. We presented a framework that is able to find 2 communities in graphs of size up to 410 vertices using only NISQ-era devices. We explored the potential of our approach as new and better QC hardware becomes available in two ways. First, we used the classical Gurobi solver [181] to simulate the performance improvements in QLS as the subproblem size is increased (see Fig. 4.2B). We generate a 2000 node random graph with realistic community structure and known modularity [216]. Unsurprisingly, QLS finds the optimal solution faster (using fewer local search iterations) as the subproblem size increases. Second, we demonstrate the need for quantum acceleration by demonstrating the limitations of existing state-of-the-art solvers. We used Gurobi [181] as a subproblem solver with subproblem size of 200. Fig. 4.2A shows that for the subproblem of this size, Gurobi cannot produce a good solution quickly. We compared Gurobi with time limit 0.25s (the running time of QA on D-Wave) with Gurobi with time limit 1000s, with the assumption that Gurobi would converge to a good solution. We use the running time of QA as our estimate because at the time of writing we do not have a good way of measuring the running time of QAOA due to the architecture of the IBM Q Experience. We expect QAOA to have similar performance. This assumes that quantum methods would scale well to larger problems, which is a strong assumption. However, the goal here is to motivate the exploration of quantum optimization heuristics by showing the limitations of classical state-of-the-art and not to demonstrate quantum advantage. Using a better solution within the local search enables 25% (4 iterations) improvement in time to convergence (convergence is defined as three iterations with

no improvement). This demonstrates that the subproblems become computationally hard even for sizes that are small enough to potentially fit on near-term devices. It is important to note that even though in our experiments Gurobi performed better than other integer programming solvers, it is quite possible that other solvers can perform better on this problem, especially after tuning. Indeed, in the past the improvements in classical heuristics have forced researchers to downgrade claims of quantum advantage [149, 150]. However, demonstrating quantum advantage is outside of scope of this Section. Instead, we use these results to motivate our hybrid approach by showing the computational complexity of the subproblems offloaded to quantum solvers. As quantum solvers improve and become capable of providing speedups at *subproblem* level, out QLS will enable us to leverage these speedups at the global problem level.

4.2 Multilevel Combinatorial Optimization Across Quantum Architectures

Across different domains, computational optimization problems that model large-scale complex systems often introduce a major obstacle to solvers even if tackled with high-performance computing systems. There are several reasons for this, including but not limited to a large number of variables and even larger number of interactions, dimensionality required to describe each variable or interaction, and time slices. The combinatorial and mixed integer optimization problems introduce additional layers of complexity with integer variables often making the problem NP-hard (e.g., in cases of non-linearity and non-convexity). A common practical approach to solve these problems is to use iterative methods. The iterative methods, while being composed with completely different algorithmic principles, share a common property: several fast improvement iterations are followed by a long tail of slow improvement iterations [252, 131]. Typically, in such iterative algorithms, solving a large-scale system with respect to the first-order interaction laws per iteration advances the solution towards a local attraction basin at each iteration, which often appears to be false with respect to the global optimal solution. In other words, local methods tend to converge to a false local optimum, which often corresponds to the solution of much lower quality than the true global optimum [113]. Moreover, in some cases, another problem may exist within each iteration – the algorithms used to solve them are not necessarily exact. To accelerate the solvers at each iteration various heuristics, parallel-friendly methods, and ad-hoc tricks are employed, which often reduce the

quality of the solution.

In this section, we take steps towards building more robust solvers for mid- to large-scale combinatorial optimization problems by fusing two areas whose simultaneous application is only beginning to be explored, namely, quantum computing and multiscale methods. Recent advances in quantum computing provide a new approach for algorithm development for many combinatorial optimization problems. However, Noisy Intermediate Scale Quantum (NISQ) devices are widely expected to be limited to a few hundred, and for certain sparse architectures up to a few thousands qubits. The current state of quantum computing theory and engineering suggests moderately optimistic expectations. In particular, it is believed that in the near future, we will witness relatively robust small-scale architectures with much less noise. This would allow algorithms like the Quantum Approximate Optimization Algorithm (QAOA) and Quantum Annealing (QA) to be run on hardware with limited error correction. Given the realistic level of precision and, in the case of QAOA, ansatz depth, these algorithms are prime candidates for demonstrating Quantum Advantage, that is solving a computationally hard problem (such as NP-hard) faster than classical state-of-the-art algorithms. Such algorithms are our first building block.

The multiscale optimization method is our second building block. These methods have been developed to cope with large-scale problems by introducing an approach to avoid entering false local attraction basins (local optima), a complementary method to stochastic and multi-start strategies that help to escape it if trapped. Because of historical reasons, on graph problems, they have been termed *multilevel* (rather than multiscale), which we will use here. The multilevel (or multiscale) methods have a long history of breakthrough results in many different optimization problems [55, 38, 159, 86, 129, 205, 92, 214, 215, 141, 103, 204, 223, 221, 210, 209] and have been implemented on a variety of hardware architectures. The success of multilevel methods for optimization problems supports our optimism about proposed ideas.

There is no unique prescription on how to design multilevel algorithms, but the main idea behind them is to "*think globally while acting locally*" on a hierarchy of coarse representations of the original large-scale optimization problem. A multilevel algorithm therefore begins by constructing such a hierarchy of progressively smaller (coarser) representations of the original problem. The goal of the next coarser level in this hierarchy is to approximate the current level problem with a coarser one that has fewer degrees of freedom and thus can be solved more effectively. When the coarse problem is solved, its solution is projected back to the finer level and further refined, a stage that is called

uncoarsening. As a result of such a strategy, the multilevel framework is often able to significantly improve the running time and solution quality of optimization methods. The quality of multilevel algorithms in large part depends on that of the optimization solvers applied at all stages of the multilevel framework. In many cases, these locally acting optimization solvers are either heuristics that get stuck in a local optimum or exact solvers applied on a small number of variables (i.e., on subproblems). In both cases, the quality of a global solution can significantly suffer depending on the quality of the solution from the local solver. The optimization algorithms running on the NISQ devices that may replace such local solvers are expected to be a critical missing component to achieve a game changing breakthrough in multilevel methods for combinatorial optimization. While the performance of these NISQ-era optimization algorithms is not fully understood (see Sec. 4.2.1.2 for an overview), in this work we do not attempt to rigorously benchmark them. Rather, we focus on the problems arising when integrating these optimization algorithms into a multilevel framework.

In this Section, we introduce Multilevel Quantum Local Search (ML-QLS), which uses an iterative refinement scheme on NISQ devices within a multilevel framework. ML-QLS extends the Quantum Local Search (QLS) (presented in Section 4.1) approach to solve larger problems. This work builds on early results using a multilevel framework and the D-Wave quantum Annealer for the Graph Partitioning Problem [246]. We demonstrate the general approach of solving combinatorial optimization problems with NISQ devices in a multilevel framework on two well-known problems as our use cases. In particular, we solve the Graph Partitioning Problem and the Community Detection Problem on graphs up to approximately $29,000$ nodes using subproblem sizes of 20 and 64 that map onto NISQ devices such as IBM Q Poughkeepsie (20 qubits) and D-Wave 2000Q ($\sim$2048 qubits). Such graphs are orders of magnitude larger than those solved by state-of-the-art hybrid quantum-classical methods. To implement this approach, we develop a novel efficient subproblem formulation method.

The rest of Section is organized as follows. In Section 4.2.1, we discuss the relevant background on quantum optimization, multilevel methods, and define the problems. In Sections 4.2.2 and 4.2.3, we discuss the hybrid quantum-classical multilevel algorithm and computational results, respectively. A discussion of the outlook and important open problems that represent major future research directions are presented in Section 4.2.4.

4.2.1 Background

The methods proposed and implemented in this work aim to solve large graph problems by integrating NISQ optimization algorithms into a multilevel scheme. In this section, we provide a brief introduction into the target graph problems (Sec. 4.2.1.1) and multilevel methods (Sec. 4.2.1.3). For an overview of quantum optimization schemes see Chapter 3.

4.2.1.1 Problem Definitions

In this Section, we reintroduce modularity maximization problem to help us highlight its connection to the graph partitioning problem. For additional details on modularity maximization (also known as community detection), see Section 3.4.1. Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. We denote by n and m the numbers of nodes and edges, respectively. For each node i, define $\mathbb{v}_i \in \mathbb{R}$ as the volume of node i and $A_{ij} \in \mathbb{R}$ as the positive weight of edge (i, j). For a fixed integer k, the *Graph Partitioning Problem* is to find a partition $V_1, \ldots, V_k$ of the vertex set V into k parts with equal total node volume such that the total weight of *cut edges* is minimized. A *cut edge* is defined as an edge whose end points are in different partitions. A requirement of equal total sizes of V_i for all i is sometimes referred as *perfectly balanced* graph partitioning, otherwise an imbalancing parameter is usually introduced to allow imbalanced partitions [45]. However, in this Section we deal with perfect balancing constraints and limit the number of parts to $k = 2$. In this case we can write the GP problem as the following quadratic program

$$\begin{aligned} \max \quad & \mathbf{s}^T A \mathbf{s} \\ \text{s.t.} \quad & \sum_{i=1}^{n} \mathbb{v}_i s_i = 0 \\ & s_i \in \{-1, 1\},\ i = 1, \ldots, n, \end{aligned} \tag{4.2}$$

which, as shown in [245], can be reformulated into the following Ising model,

$$\begin{aligned} \max \quad & \mathbf{s}^T (\beta A - \alpha \mathbb{v}\mathbb{v}^T) \mathbf{s} \\ \text{s.t.} \quad & s_i \in \{-1, 1\},\ i = 1, \ldots, n, \end{aligned} \tag{4.3}$$

for some constants $\alpha, \beta > 0$, where $\mathbb{v}$ is a column vector of volumes such that $(\mathbb{v})_i = \mathbb{v}_i$.

Maximization of modularity is a famous problem in network science where the goal is to

find communities in a network through node clustering (also known as modularity clustering) [176]. For the graph G, the problem of Modularity Maximization is to find a partitioning of the vertex set into one or more parts (communities) that maximizes the modularity metric. The modularity matrix is a symmetric matrix given by

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2|E|}, \tag{4.4}$$

where k_i is the weighted degree of node i, namely, $k_i = \sum_j A_{ij}$. Whereas the modularity is typically defined on unweighted graphs, within the multilevel framework, due to the coarsening of nodes, we primarily work with weighted graphs. It can equivalently be written in matrix-vector notation as

$$B = A - \frac{1}{2|E|}\mathbb{k}\mathbb{k}^T \tag{4.5}$$

where $\mathbb{k}$ is a vector of weighted degrees of the nodes in the graph. For up to 2 communities, the *Modularity Maximization Problem*, also referred to as the *Community Detection Problem*, can be written in Ising form as follows:

$$\begin{aligned} \max \quad & \frac{1}{4|E|}\mathbf{s}^T\Big(A - \frac{1}{2|E|}\mathbb{k}\mathbb{k}^T\Big)\mathbf{s} \\ \text{s.t.} \quad & s_i \in \{-1, 1\},\ i = 1, \ldots, n \end{aligned} \tag{4.6}$$

where the objective value of equation 4.6, for a given assignment of resulting communities, is referred to as the *modularity*. For more than 2 communities, the Ising formulation of the Community Detection Problem is given in [172].

Note that the above formulation of Modularity Maximization can be viewed as the Graph Partitioning Problem in the Ising model given in equation (4.3) where the volume of a node is defined as the weighted degree and the penalty constants $\beta = 1, \alpha = \frac{1}{2|E|}$. We exploit this deep duality between the two problems in our implementation.

4.2.1.2 On the Scalability of Quantum Optimization Heuristics

The question of asymptotic scaling is the central question in the analysis of algorithms. Unfortunately, for many of the most promising quantum optimization algorithms, rigorous analysis (such as provable approximation ratios) beyond the most simple problems is unavailable. Therefore

the researchers have to resort to experimental evaluations and back-of-the-envelope projections. This gives rise to the second major complication, namely, the fact that empirical results on small problem instances are almost totally uninformative of the overall scaling behavior. Famously, the adiabatic quantum algorithm initially appeared to be practically useful for solving NP-complete problems in polynomial time based on numerical simulations of problems of up to tens of variables [74, 72]. However, later analysis has shown that for many problems, both synthetica ones that are classically easy and hard problems like 3-SAT [247], eigengaps diminish exponentially, leading to exponential worst-case running time for the adiabatic quantum algorithm [11]. This provides a cautionary example for researchers trying to analyze modern quantum optimization approaches. The exact number of variables needed for the separation between polynomial and exponential scaling to become apparent varies from problem to problem, and the separation might not be clear for small problem instances (i.e. the scaling that is in fact exponential looks polynomial for small instance sizes). However, the increased size of the simulations as well as the hardware (in the case of quantum annealers, reaching into thousands of qubits) provides increasing confidence in the potential of quantum optimization heuristics.

Performance on the D-Wave quantum annealer depends on the input, the solver, and the hardware platform [156]. Changes to the solver include modifying the anneal time or schedule. In this case the platform is the D-Wave 2000Q. Bigger problems are better when comparing against classical approaches. Pre-processing strategies such as variable reduction, error mitigation and improved embeddings applied to the input contribute to more optimized performance. Minimizing the number of qubits per variable results in shorter chains and improved embeddings. The solver tuning strategies include finding an optimal anneal time dependent on the problem, as well as modifying the anneal schedule. Longer anneal times may be required for larger problems. Performance scaling for quantum applications is viewed as problem size vs. time to solution (TTS) for optimized run parameters, similar to classical.

Considering the ground-state success probability for Sherrington-Kirkpatrick (SK) model and MAX-CUT problems of increasing size was shown to be helpful in understand scaling [104]. When evaluating the TTS it is important to optimize the run parameters as much as possible, in particular the optimal annealing time. Small problems require short anneals where the success probability is always close to unity and insensitive to the annealing time. Larger problems require long anneals where the success probability dominates.

Work focused on determining the optimal anneal time on a quantum annealer over classical simulated annealing (SA) for logical-planted instances demonstrated a scaling advantage over SA on the D-Wave quantum annealer [10]. Scaling advantage in both system size and inverse temperature was demonstrated for simulation of frustrated magnetism in quantum condensed matter for the D-Wave quantum annealer over classical path-integral Monte Carlo (PIMC) [135].

A D-Wave API is available for collecting timing information on the details of the Quantum Processing Unit (QPU) access time [62]. The QPU access time consists of programming time, sampling time, post-processing time, and a small amount of time spent in initialization. Programming time measures how long it takes to initialize the problem on the QPU. The sampling time is further broken down into per sample times for the anneal, readout, and delay. Using this API, runtime scaling for the quantum isomer search problem on the D-Wave QA was shown to grow linearly with problem size [242]. In this case the problem size is defined by the number of carbon atoms n in an alkane, which translates to $4(n-2)$ variables. And the number of isomer solutions increases with the number of carbon atoms. We use this API similarly to show the scaling of the D-Wave contribution on problems of increasing size in Sec. 4.2.3.1.

The scaling of Quantum Approximate Optimization Algorithm (QAOA) is particularly hard due to two factors, namely the presence of outer-loop optimization and the lack of understanding of the scaling of the depth of QAOA circuit required to achieve a good solution. The first factor complicating the analysis of QAOA is the outer-loop optimization, i.e., the need to optimize parameters $\boldsymbol{\beta}, \boldsymbol{\gamma}$ in Eq. 3.1. As the landscape is known to be highly non-convex [224, 220, 267], this optimization becomes a daunting task. Little is known about the structure of this landscape, making it hard to provide an upper bound on the computational hardness of the problem. At the time of writing the best known upper bound is that the problem of optimizing $\boldsymbol{\beta}, \boldsymbol{\gamma}$ is as hard as finding a global minimizer to a nonconvex problem, where even verifying a feasible point is a local minimizer is NP-hard [166]. However, in practice a number of techniques have been developed to successfully solve this problem. While the structure is hard to analyze, there have been successful attempts to leverage it using machine learning techniques [83, 250, 133]. These results make us hopeful that with the help of a pre-trained model, high-quality QAOA parameters can be found in a small number of objective evaluations. There have been promising results showing that in higher-depth regime for some problems it is possible to avoid optimization altogether and use a smoothly extrapolated set of parameters reminiscent of adiabatic schedule [267].

The second factor in the lack of analytical and empirical results on QAOA behavior in low- to medium-depth regime (e.g., $5 \leq p \leq 100$). Analytically, QAOA appears to be hard to analyze beyond $p = \{1, 2\}$ for non-trivial problems [256, 36, 238]. At the same time, even for very simple problems and small instance sizes it is clear that achieving a good solution requires going beyond at least $p = 5$ [220, 59, 238]. Therefore we have to rely on empirical results to answer the question of exactly how large the p needs to be in order to achieve a good solution. This empirical evaluation is impeded by the complexity of simulating QAOA in medium depths. On one end, traditional state-vector based simulators have running time exponential in number of *qubits*, limiting the problem sizes to tens of variables. On the other end, tensor network based simulators have running time that is exponential in the number of *gates*, limiting the depth of the QAOA circuit that can be simulated. At the time of the writing, the state-of-the-art simulators were limited to simulating a thousand qubits to depth $p = 5$ [115]. These two constraints (on number of qubits for one simulation approach and on the depth for the other) make it challenging to numerically analyze QAOA performance in the crucial zone of medium-sized problems (hundreds to thousands of variables) and medium-depth circuit ($p > 10$). The high levels of noise and small number of available qubits make this analysis impossible on the currently available quantum hardware. All of the aforementioned complications contribute to the lack of the results showing how QAOA depth p scales with the size of the problem. At the same time, results presented in [59, 267] indicate that at least for the problem sizes small enough to fit on near-term quantum computers, $10 \leq p \leq 30$ is sufficient to achieve high-quality solutions.

Due to the limitations of the available hardware, in this work we do not run full QAOA ansatz on the IBM Q hardware. Instead, we use a shallow-depth hardware-efficient ansatz (HEA) [123]. Much less is known about the potential of such ansatzes to produce quantum speedup. Numerical experiments on small problem instances do not show that quantum entanglement provides an advantage in optimization [169]. At the same time, recent analytical results show that HEAs are efficiently simulatable classically in constant depth [170] and suffer from exponentially vanishing gradients in polynomial depth [155] (here depth is a function of number of qubits). A recent result shows that in logarithmic depth regime gradient vanishes only polynomially, making HEAs trainable in this regime [49]. This result indicates a potential for quantum advantage using HEAs in logarithmic depth, as they are both hard to simulate classically and do not suffer from exponentially vanishing gradients. Evaluating the potential for quantum advantage from using logarithmic-depth

HEAs in QAOA setting for optimization remains an open problem.

4.2.1.3 Multilevel Combinatorial Optimization with NISQ solvers

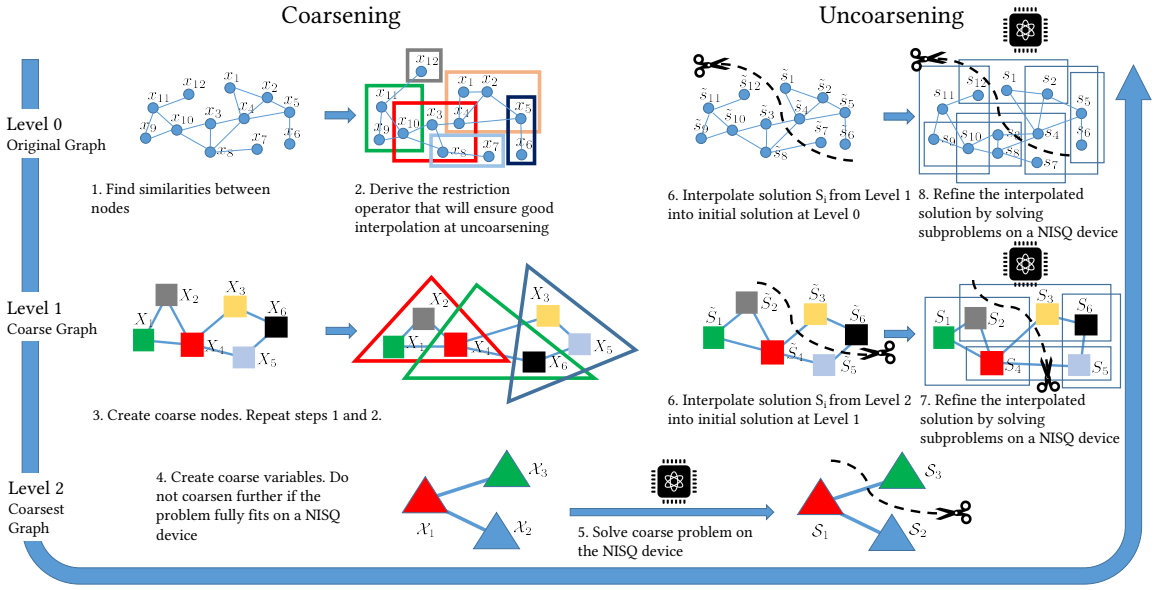


Figure 4.3: V-cycle for a graph problem. First, the problem is iteratively coarsened (left). Second, the coarse problem is solved using a NISQ optimization solver (bottom). Finally, the problem is iteratively uncoarsened and the solution is refined using a NISQ solver (right).

The goal of the multilevel approach for optimization problems on graphs is to create a hierarchy of coarsened graphs G_0, G_1, ... ,G_k in such a way that the next coarser graph G_{i+1} "approximates" some properties of G_i (that are directly relevant to the optimization problem of interest) with fewer degrees of freedom. After constructing such a hierarchy, the coarsening is followed by solving the problem on G_k as best as we can (preferably exactly) do and finally uncoarsening the solution back to G_0 through gradual refinement at all levels of the hierarchy, with a refined solution at level $i+1$ serving as the initial solution at level i. The entire coarsening-uncoarsening process is called a V-cycle. There are other variations of hierarchy levels' coarsening-uncoarsening order, e.g., W- and Full cycles [39]. Fig. 4.3 presents an outline of a V-cycle.

Typically, when solving problems on graphs in which nodes represent the optimization variables (such as those in the partitioning and community detection), having fewer degrees of freedom implies a decreased number of nodes in each next coarser graph $|V_0| > |V_1| > |V_2| > ... >$

$|V_k|$.[1] With a smaller number of variables at each level, one can use more sophisticated algorithms at each level. However, it is still not sufficient to solve the whole problem until the coarsening reaches the coarsest level. As a result, at each level, the actual solution is produced by a refinement. Refinement is typically implemented with a decomposition method that uses a previous iteration or a coarser level solution as an initial guess. The multilevel algorithms rely heavily [254] on the quality of refinement solvers for small and local subproblems at all levels of coarseness. Thus, the most straightforward way to use NISQ devices in multilevel frameworks is to iteratively apply them as local solvers to refine a solution inherited from the coarse level. Because the refinement is executed at all levels of coarseness, it is clear that *even a small improvement of a solution at the coarse level may cause a major improvement at the finest scale*. Typically, this is the most time-consuming stage of the multilevel solvers which is expected to be fundamentally better if improved by NISQ devices.

Most refinement solvers in multilevel frameworks rely on fast but low-quality heuristics, rather than on the ability to compute an optimal solution. Moreover, in many existing solvers, the number of variables in such local subproblems is comparable with or smaller than the the size of the problems that can be directly embedded on the NISQ devices (see examples in [103, 141]), making them a perfect target for NISQ optimization algorithms. In most multilevel/multiscale/multigrid-based optimization solvers, a refinement consists of covering the domain (or all variables) with *small* subsets of variables (i.e., small subproblems) such that solving a small local problem on a subset improves the global solution for the current level.

Multilevel graph partitioning and community detection algorithms are examples of the most successful applications of multilevel algorithms for large graphs, achieving excellent time/quality trade-off [45]. In this work, we use the simplest version of coarsening (in order to focus on the hybrid quantum-classical refinement) in which the edges of the fine level graph are collapsed and create coarse level vertices by merging the fine level ones. There are several classes of refinement for both problems but in all of them, at each step a small subset of nodes (or even a singleton) is reassigned with partition (or cluster) that either better optimizes the objective or improves constraints. Some variants of stochastic extensions also exist.

[1] Note that this does not necessarily imply $|E_0| > |E_1| > |E_2| > ... > |E_k|$

4.2.1.4 On Scalability of Multilevel Methods

If we do not consider algorithmic frameworks with very limited space (such as streaming), the scalability of a traditional multilevel framework with its instance graphs is practically limited to the available memory size. Requirements to keep graphs in memory (not necessarily RAM) for multilevel partitioning and community detection are similar to those of matrices for multigrid [39], so the complexity is also comparable up to the factor of refinement. Theoretically, the multilevel and multigrid frameworks exhibit $O(|E|)$ or O(number of non-zeros in a matrix) complexity. However, for optimization on graphs, the refinement stage is typically computationally more expensive than that for multigrid (e.g., compare Gauss-Seidel relaxation sweeps [143] and Kernighan-Lin or min-cut/max-flow refinement in graph partitioning [45]) because at each step of the refinement, an integer problem has to be solved. Refinement for the relaxed versions of integer problems (e.g., for the minimum 2-sum or bandwidth [214]) are usually faster but the quality suffers and in the end they should follow some rounding scheme for integer solution. However, even if the the complexity of refinement is linear in the number of edges or corresponding matrix non-zeros, some overhead is typically introduced for the integer problems.

In this work, we use a simple coarsening model which folds edges by merging pairs of nodes. The situation with the scalability of multilevel frameworks if high-order interpolation coarsening is involved (e.g., algebraic multigrid inspired weighted aggregation [213] when nodes can be split into several fractions, and different fractions form coarse nodes) is different. The high-order interpolation coarsening may result in increasing number of edges at several fine levels immediately implying increased running time. In such cases, the complexity can increase to $\max_{\text{level } i} O(E_i)$. Subsequently, a larger graph at each level requires more intensive refinement. In addition, the number of refinement calls required to achieve a very good solution strongly depends on the coarsening quality which makes it difficult to get a complexity required for a nearly optimal solution. The only practical solution to that is artificially limiting the number of refinement calls (see all major solvers such as Metis, Jostle, Kahip, and Scotch reviewed in [45]).

The criteria for limiting the number of refinement calls is never ideal. The refinemnt algorithms always heuristically decide what vertex or group of vertices should be optimized with respect to the current assignment of vertices to clusters (or parts) to make the current solution better. Typically, they take so called "boundary" nodes in all parts, i.e., those whose move from part to part

can potentially improve the solution and optimize them or their groups. Therefore, the scalability of multilevel graph partitioning and clustering frameworks can be loosely described in terms of $|V|$ and $|E|$. Instead, it is better to describe it in terms of expected cut similar to the analysis in [128]. To the best of our knowledge, there is no analysis that connects the size of the graph and performance of multilevel algorithms providing any practically useful theoretical bounds. Currently, the best way to describe the scalability of multilevel graph partitioning and clustering solvers is to use $O(|E|)$. However, the hidden constant in the linear complexity depends on the type of refinement. Thus, we anticipate that with the advent of reliable quantum hardware, one can expect a significant improvement in the running time and quality of the refinement in multilevel frameworks which will eliminate computationally expensive solvers to locally optimize groups of variables. We refer the reader to one of the most recent examples of multilevel scalability in [65] (e.g., see Table IV) in which a graph with 3.8B edges was (suboptimally) solved in 255 seconds.

4.2.2 Methods

An iterative improvement scheme is a common approach for solving large scale problems with NISQ devices. Traditionally, this is done by formulating the entire problem in the Ising model or as a QUBO and then solving it using hybrid quantum-classical algorithms (see, for example, "qbsolv" from D-Wave systems [34]). These methods decompose the large QUBO into smaller sub-QUBOs or decrease the number of degrees of freedom to fit the subproblem on the hardware (for example, using a multilevel scheme), and iteratively improve the global solution by solving the small subproblems (sub-QUBOs). One of the main limitations of this approach is the size and density of the original QUBO. For example, in the graph partitioning formulation given by equation 4.3, the term $\mathbb{v}\mathbb{v}^T$ leads to the formulation of a completely dense $n \times n$ QUBO matrix regardless of whether or not the original graph was sparse. Storing and processing this dense matrix can easily make this method prohibitively computationally expensive even for moderately sized problems. In our implementation of Quantum Local Search (QLS) [226] we circumvent this limitation by developing a novel subproblem formulation of the Graph Partitioning Problem and Modularity Maximization as a QUBO that does not require formulating the entire QUBO.

Another concern is the effectiveness of selection criteria of candidate variables (or nodes) to be included in each subproblem. A common metric used in selecting whether or not a variable is to be included in the subproblem is whether or not changing the variable value would reduce (increase)

the objective value for a minimization (maximization) problem. Thus, since computing the change in objective value for a small change in the solution is performed multiple times, it is important to ensure that this computation is efficient. We derive a novel efficient way to compute the change in the objective value of the entire QUBO also without formulating the entire QUBO and thus provide an efficient refinement scheme using current NISQ devices.

We begin by introducing an efficient QUBO subproblem formulation for the Graph Partitioning Problem, and the Community Detection Problem. Then we present an efficient way to compute the gain and change in the objective of the entire QUBO. Finally, we put it all together and outline our algorithm.

4.2.2.1 QUBO formulation for subproblems

Let M be an $n \times n$ symmetric matrix that represents the QUBO for a large scale problem such that it is prohibitively expensive to either generate or store M. However, for QLS we need to generate constant-size sub-QUBOs of M which in turn represent subproblems of the original problem. In order to generate a sub-QUBO, let k be the size of the desired sub-QUBO. In other words, the sub-QUBO will have k variables and $n - k$ *fixed variables* that remain invariant for this specific sub-QUBO. We refer to the k variables as *free variables*. Without loss of generality, let the the first k variables of $\mathbf{s}$ be the free variables, then we write $\mathbf{s}$ as

$$\mathbf{s} = \begin{bmatrix} \mathbf{s}_v \\ \mathbf{s}_f \end{bmatrix},$$

where $\mathbf{s}_v$ represents the k free variable terms and $\mathbf{s}_f$ represents the $n - k$ fixed terms. In the next step, M can be represented using block form

$$M = \left[\begin{array}{c|c} M_{vv} & M_{vf} \\ \hline M_{vf}^T & M_{ff} \end{array}\right] \tag{4.7}$$

such that M_{vv} is a $k \times k$ matrix. Next, we can write $\mathbf{s}^T M \mathbf{s}$ as

$$\mathbf{s}^T M \mathbf{s} = \mathbf{s}_v^T M_{vv} \mathbf{s}_v + \mathbf{s}_v^T (2 M_{vf} \mathbf{s}_f) + \mathbf{s}_f^T M_{ff} \mathbf{s}_f \tag{4.8}$$

Since $\mathbf{s}_f$ are fixed values, we have $\mathbf{s}_f^T M_{ff} \mathbf{s}_f$ as a constant thus

$$\min\ \mathbf{s}^T M \mathbf{s} = \min\ \mathbf{s}_v^T M_{vv} \mathbf{s}_v + \mathbf{s}_v^T (2 M_{vf} \mathbf{s}_f) \tag{4.9}$$

From equation (4.7), we have

$$\mathbb{v}\mathbb{v}^T = \left[\begin{array}{c|c} \mathbb{v}_v \mathbb{v}_v^T & \mathbb{v}_v \mathbb{v}_f^T \\ \hline \mathbb{v}_f \mathbb{v}_v^T & \mathbb{v}_f \mathbb{v}_f^T \end{array}\right] \tag{4.10}$$

Therefore, from equation (4.9), we have

$$\min\ \mathbf{s}^T \mathbb{v}\mathbb{v}^T \mathbf{s} = \min\ \mathbf{s}_v^T \mathbb{v}_v \mathbb{v}_v^T \mathbf{s}_v + 2 \mathbf{s}_v^T \mathbb{v}_v \mathbb{v}_f^T \mathbf{s}_f \tag{4.11}$$

The formulation in (4.11) is particularly important because it shows that the matrix $\mathbb{v}\mathbb{v}^T$ does not need to be explicitly created at each iteration during refinement. This is a crucial observation because $\mathbb{v}\mathbb{v}^T$ is a completely dense matrix.

As described in Sec. 4.2.1.1, the Community Detection Problem is given by

$$\max\ \frac{1}{4|E|} \mathbf{s}^T \Big(A - \frac{1}{2|E|} \mathbb{k}\mathbb{k}^T \Big) \mathbf{s} \tag{4.12}$$

or

$$\min\ \mathbf{s}^T \Big(\frac{1}{2|E|} \mathbb{k}\mathbb{k}^T - A \Big) \mathbf{s} \tag{4.13}$$

and the Graph Partitioning Problem is given by

$$\min\ \mathbf{s}^T \Big(\alpha \mathbb{v}\mathbb{v}^T - \beta A \Big) \mathbf{s}. \tag{4.14}$$

In the above formulation, modularity clustering can be viewed as the Graph Partitioning Problem in a QUBO model, where the volume of a node is defined as the weighted degree and the penalty constant is $\frac{1}{|E|}$ Therefore, in both cases we can perform a refinement while defining fixed values as

$$\min\ \mathbf{s}^T\Big(\frac{1}{2|E|}\mathbb{k}\mathbb{k}^T - A\Big)\mathbf{s} = \min\ \mathbf{s}_v^T\Big(\frac{1}{2|E|}\mathbb{k}_v\mathbb{k}_v^T\Big)\mathbf{s}_v + \mathbf{s}_v^T\Big(\frac{1}{|E|}\mathbb{k}_v\mathbb{k}_f^T\Big)\mathbf{s}_f - \mathbf{s}^T A\mathbf{s} \tag{4.15}$$

and

$$\min\ \mathbf{s}^T\Big(\alpha\mathbb{v}\mathbb{v}^T - \beta A\Big)\mathbf{s} = \min\ \mathbf{s}_v^T\Big(\alpha\mathbb{v}_v\mathbb{v}_v^T\Big)\mathbf{s}_v + \mathbf{s}_v^T\Big(2\alpha\mathbb{v}_v\mathbb{v}_f^T\Big)\mathbf{s}_f - \beta\mathbf{s}^T A\mathbf{s} \tag{4.16}$$

with

$$\min\ -\beta\mathbf{s}^T A\mathbf{s} = \min\ -\beta\mathbf{s}_v^T A_{vv}\mathbf{s}_v - \mathbf{s}_v^T(2\beta A_{vf}\mathbf{s}_f) \tag{4.17}$$

The formulation in (4.15) and (4.16) are particularly important during the refinement step because this implies that the complete dense (and therefore prohibitively large) QUBO or Ising model does not need to be created at each iteration. These formulations also demonstrate a close relationship between the Graph Partitioning Problem and the Community Detection Problem.

4.2.2.2 Efficient Evaluation of the Objective

In order to select the free variables for the subproblem, we need to be able to efficiently compute the change of the objective function by moving one node from one part to another. In other words, for each vertex v, we need to efficiently compute the *gain* which is the decrease (or increase) in the edge-cut together with penalty if v is moved to the other part.

For a symmetric matrix M, the change in the value $Q = \mathbf{s}^T M\mathbf{s}$ by flipping a single variable s_i corresponding to the node i is given by

$$\Delta Q(i) = 2(\sum_{j\in C_1} M_{ij} - \sum_{j\in C_2} M_{ij}) \tag{4.18}$$

where C_1 and C_2 correspond to all variables with $s_i = -1$ and $s_i = 1$ respectively. Next, we define

$$deg(v, C) := \sum_{j\in C} A_{vj}; \quad Deg(C) := \sum_{i\in C} k_i; \quad Vol(C) := \sum_{i\in C} \mathbb{v}_i$$

then

$$2(\sum_{j\in C_1} A_{ij} - \sum_{j\in C_2} A_{ij}) = 2deg(v_i, C_1) - 2deg(v_i, C_2)$$

and finally

$$\begin{aligned} 2(\sum_{j\in C_1} (\mathbb{v}\mathbb{v}^T)_{ij} - \sum_{j\in C_2} (\mathbb{v}\mathbb{v}^T)_{ij}) &= 2\Big(\mathbb{v}_i \sum_{j\in C_1, i\neq j} \mathbb{v}_j - \mathbb{v}_i \sum_{j\in C_2} \mathbb{v}_j\Big) \\ &= 2\mathbb{v}_i\big(Vol(C_1\backslash i) - Vol(C_2)\big) \end{aligned}$$

where we assume that $i \in C_1$. This expression can be computed in $O(1)$ time.

In the same way

$$\begin{aligned} 2(\sum_{j\in C_1} (\mathbb{k}\mathbb{k}^T)_{ij} - \sum_{j\in C_2} (\mathbb{k}\mathbb{k}^T)_{ij}) &= 2\Big(k_i \sum_{j\in C_1, i\neq j} k_j - k_i \sum_{j\in C_2} k_j\Big) \\ &= 2k_i\big(Deg(C_1\backslash i) - Deg(C_2)\big) \end{aligned}$$

can also be computed in $O(1)$ time given $Deg(C_1)$ and $Deg(C_2)$, where $Deg(C_i)$ represents the sum of weighted degrees of nodes in community i.

Therefore, the change in modularity is given by

$$\Delta Q(i) = \frac{k_i}{|E|}\big(Deg(C_1\backslash i) - Deg(C_2)\big) - 2\Big(deg(v_i, C_1) - deg(v_i, C_2)\Big) \tag{4.19}$$

and change in edge-cut together with penalty value is given by

$$\Delta Q(i) = 2\alpha\mathbb{v}_i\big(Vol(C_1\backslash i) - Vol(C_2)\big)) - 2\beta\Big(deg(v_i, C_1) - deg(v_i, C_2)\Big) \tag{4.20}$$

For each node i, both expressions (4.19) and (4.20) can be computed in $O(k_i)$ time, where k_i is the unweighted degree of i.

At no point during the algorithm should the complete QUBO matrix be formulated. This also applies to the process of evaluating a given solution. In other words, evaluating the modularity for the Community Detection Problem or edge-cut together with penalty term for the Graph

Partitioning Problem should be done in $O(1)$ time and space. The term is

$$\mathbf{s}^T \mathbb{v}\mathbb{v}^T \mathbf{s} = \big(Vol(C_1) - Vol(C_2)\big)^2$$

where as

$$\mathbf{s}^T A\mathbf{s} = 2(|E| - 2cut).$$

Therefore,

$$\mathbf{s}^T(\alpha \mathbb{v}\mathbb{v}^T - \beta A)\mathbf{s} = \alpha\Big(Vol(C_1) - Vol(C_2)\Big)^2 - 2\beta(|E| - 2cut) \tag{4.21}$$

and

$$\mathbf{s}^T\Big(\frac{1}{2|E|}\mathbb{k}\mathbb{k}^T - A\Big)\mathbf{s} = \frac{1}{2|E|}\Big(Deg(C_1) - Deg(C_2)\Big)^2 - 2(|E| - 2cut) \tag{4.22}$$

where equations (4.21) and (4.22) give the formulations for computing the modularity and edge-cut with corresponding penalty value respectively without creating the QUBO matrix.

4.2.2.3 Algorithm Overview

Now we can combine the building blocks described in the previous two subsections. Let $G = (V, E)$ be the problem graph. ML-QLS begins by coarsening the problem graph. During the coarsening stage, for some integer k, a hierarchy of coarsened graphs $G = G_0, G_1, \ldots, G_k$ is constructed. In this work, we used the coarsening tools implemented in KaHIP Graph Partitioning package [217]. We used the coarsening implementation that is performed using maximum weight matching with "expansion^{*2}" metric as described in [112]. The maximum edge matching is found using the Global Path Algorithm [112]. In the next step, a QUBO is formulated for the smallest graph G_k and solved on the quantum device. If $|V_k|$ is greater than the hardware size[2], QLS [226] with a random initialization is used to solve for G_k. Then, the solution is iteratively projected onto finer levels and refined using QLS. The algorithm overview is presented in Alg. 2.

For the Graph Partitioning Problem, the initial weight of each node is one by definition, therefore coarsening of the nodes keeps the total node volume constant at each coarsening level. For the Community Detection Problem, the initial weight of each node is set to the degree of the node. This ensures that the size of the graph (total number of weighted edges) is also kept constant at

[2]more specifically, greater than the maximum number of variables in a problem that can be embedded on the device

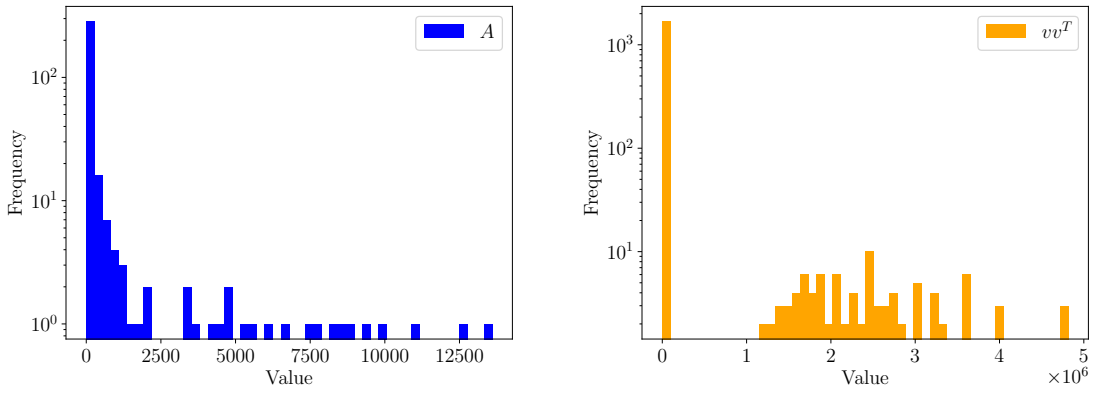
Algorithm 2 Multilevel Quantum Local Search

```
function ML-QLS(G, problem_type):
    if problem_type is modularity then
        G = UpdateWeights(G)
    end
    G_0, G_1, ..., G_k = KaHIPCoarsen(G)
    if |V_k| ≤ HardwareSize then
        // solve directly
        QUBO = FormulateQUBO(G_k)
        solution = SolveSubproblem(QUBO)
    else
        // use QLS
        initial_solution = RandomSolution(G_k)
        solution = RefineSolution(G_k, initial_solution)
    end
    for G_i in G_{k-1}, G_{k-2}, ..., G_0 do
        projected_solution = ProjectSolution(solution, G_i, G_{i+1})
        solution = RefineSolution(G_i, projected_solution)
    end
    return solution
function RefineSolution(G_i, projected_solution):
    solution = projected_solution
    while not converged do
        ΔQ = ComputeGains(G_i, solution)
        X = HighestGainNodes(ΔQ)
        QUBO = FormulateQUBO(X)
        // using IBM UQC or D-Wave QA
        candidate = SolveSubproblem(QUBO)
        if candidate > solution then
            solution = candidate
        end
    end
    return solution
```

each level. Note that Graph Partitioning is defined with respect to total node volume ($|V|$), while modularity is defined with respect to the size ($|E|$, the total number of weighted edges) of the graph.

4.2.2.4 Addressing the Limited Precision of the Hardware

One of the subproblem solvers we used in this work is Quantum Annealing, which we ran on the LANL D-Wave 2000Q machine. The D-Wave 2000Q is an analog quantum annealer with limited precision. In this work, we used a simple coarsening that constructs coarser graphs by aggregating nodes at a finer level to become a single node at the coarser level (i.e. many nodes on the finer level are merged into one node at the coarser level, with the volume of the new node set to be the sum of the volumes of the nodes on the coarser level). This causes the precision required to describe the node



(a) Edge weights of the coarsest graph `SSS12`.

(b) Entries of the matrix $\mathbb{v}\mathbb{v}^T$.

Figure 4.4: In Figure 4.4a, the maximum value is approximately 13×10^3.In Figure 4.4b, the maximum value is approximately 5×10^6 and minimum value 1. A naive scaling of QUBO matrix $A - \mathbb{v}\mathbb{v}^T$ can result in values that are too large to be handled by the quantum annealer due to its limited precision. Such values of A are ignored, leading to random balanced partitions.

volumes and edge weights for coarser graphs to increase dramatically, especially for the large scale problems. Thus, a QUBO describing the coarsest graph could require significantly more precision to represent compared to the finest graph. For example, in Graph Partitioning where the QUBO problem to be minimized is $A - \alpha\mathbb{v}\mathbb{v}^T$, the range of values in the matrix A increase at a different rate than the range of values in the matrix $\mathbb{v}\mathbb{v}^T$ during the coarsening process, increasing the precision required to describe the overall QUBO formed at each level (see an example on Fig. 4.4a). Thus, if the QUBO $A - \alpha\mathbb{v}\mathbb{v}^T$ is directly scaled to accommodate the limited precision of the device, the quality of the results can suffer. In our experiments, we observe that directly scaling the QUBO returned feasible, but low quality solutions. In order to overcome this challenge, for the problems solved on the D-Wave device, we first scaled the matrices A and $\alpha\mathbb{v}\mathbb{v}^T$ separately, and then formed the QUBO to be optimized. This approach then resulted in achieving results with high quality solutions on the D-Wave device.

4.2.3 Experiments and Results

Implementation The general framework for ML-QLS is implemented in Python 3.7 with NetworkX [101] for network operations. We have used the coarsening algorithms available in the KaHIP Graph Partitioning package [217] which are implemented in C++. The code for the general ML-QLS

framework is available on GitHub [3].

Systems The refinement algorithms presented in this work require access to NISQ devices capable of solving problems formulated in the Ising model. To this end, we have used the D-Wave 2000Q quantum annealer located at Los Alamos National Laboratory, as well as IBM's Poughkeepsie 20 qubit quantum computer available on the Oak Ridge National Laboratory IBM Q hub network together with the high-performance simulator, IBM Qiskit Aer Simulator [12]. However, our framework is modular and can easily be extended to utilize other novel quantum computing architectures as they become available.

The D-Wave 2000Q is the state-of-the-art quantum annealer at this time. It has up to 2048 qubits which are laid out in a special graph structure known as a Chimera graph. The Chimera graph is sparse, thus the device has sparse connectivity. Fully connected graphs as dense problems need to be embedded onto the device, which leads to the maximum size of 64 variables. We have used the embedding algorithm described in [35] to calculate a complete embedding of the 64 variable problem. We found this embedding only once and reused it during our experiments. We utilized D-Wave's Solver API (SAPI) which is implemented in Python 2.7, to interact with the system. The D-Wave system is intrinsically a stochastic system, where solutions are sampled from a distribution corresponding to the lowest energy state. For each subproblem, the best solution out of 10,000 samples is returned. The annealing time for each call to the D-Wave system was set to 20 microseconds.

In order to solve problems formulated in the Ising model on IBM's Poughkeepsie quantum computer and simulator, we implemented QAOA using the SBPLX [208] optimizer to find the optimal variational parameters. We allowed 2,000 iterations for SBPLX to find optimal parameters for QAOA. At each iteration, the circuit is executed 5,000 times (5,000 "shots") to obtain the statistic on the objective function. After the optimal parameters are found, the solution corresponding to best of 5,000 samples produced by running the ansatz with optimal parameters is returned. Due to the limitations of NISQ devices available in IBM Q hub network [225], we used the RYRZ variational form [57] (also known as a hardware-efficient ansatz) as the ansatz for our QAOA implementation. For the experiments run on IBM quantum device Poughkeepsie, we perform the variational parameter optimization on the simulator locally and run QAOA on the device via the IBM Q Experience cloud API. This is done due to the job queue limitations provided via the IBM Q Experience. However,

we expect to be able to run QAOA variational parameter optimization fully on a device as more devices are becoming available on the cloud. We have used GNU Parallel [240] for the large-scale numerical experiments performed on the quantum simulator.

Considering the fact that solutions from the NISQ devices and simulator do not provide optimality guarantees, we have also solved various subproblems formulated in the Ising model using the solver Gurobi [181] together with modeling package Pyomo [108]. The results using Gurobi as a solver for each subproblem are denoted as "Optimal" in our plots. Note that while each subproblem was solved and proven to be optimal for subproblem size 20, the same is not always true for subproblem size 64. For subproblem size 64 we occasionally observe non-zero gaps.

Instances A summary of the graphs used in the experiments together with their properties is presented in Table 4.1. For the Graph Partitioning Problem, we evaluate ML-QLS on five graphs, four of which are drawn from The Graph Partitioning Archive [233] (`4elt`, `bcsstk30`, `cti` and `data`) and one from the set of hard to partition graphs (`vsp_msc10848_300sep_100in_1Kout`, denoted in figures as `SSS12`) [215]. For the Modularity Maximization Problem, we evaluate ML-QLS on six graphs. The graphs `roadNet-PA-20k` and `opsahl-powergrid` are real-world networks from the KONECT dataset [136]. Graphs `msc23052` and `finan512-10k` are taken from the graph archive presented in [212]. The graphs `finan512-10k` and `roadNet-PA-20k` are reduced to 10,000 and 20,000 nodes respectively by performing a breadth-first search from the median degree node. Note that due to the high diameter of these networks and their structure (portfolio optimization problem and road network), this preserves their structural properties. `GirvanNewman` is a synthetic graph generated using the model introduced by Girvan and Newman (GN) [89]. The graph `lancichinetti1` is a synthetic graph generated using a generalization of the GN model that allows for heterogeneity in the distributions of node degree and community size, introduced by Lancichinetti et al. [137]. Table 4.2 shows the parameters used to generate the synthetic graphs.

Experimental Setup Our experiments are performed in order to compare the solutions from ML-QLS with those of high-quality classical solvers, and the best known results, if available. For the Graph Partitioning Problem, the results are compared to those produced by KaHIP [217] which is a state-of-the-art multilevel Graph Partitioning solver. The best known results are taken at The Graph Partitioning Archive [233] where applicable. In order to make our approach more comparable

Network name	$\|V\|$	$\|E\|$	d_{avg}	$d_{\max}$
SSS12	21996	1221028	111.02	722
4elt	15606	45878	5.88	10
bcsstk30	28924	1007284	69.65	218
cti	16840	48232	5.73	6
data	2851	15093	10.59	17
roadNet-PA-20k	20000	26935	2.69	7
opsahl-powergrid	4941	6594	2.67	19
msc23052	5722	103391	36.14	125
finan512-10k	10000	28098	5.62	54

Table 4.1: Properties of the networks used to evaluate ML-QLS. d_{avg} is average degree, $d_{\max}$ is maximum degree

Network name	$\|V\|$	$\|E\|$	d_{avg}	$d_{\max}$	γ	β	μ
GirvanNewman	10,000	75,000	15.0	15	1	1	0.1
lancichinetti1	10,000	76,133	15.22	50	2	1	0.1

Table 4.2: Properties of synthetic networks used in the Modularity evaluation. d_{avg} is average degree, $d_{\max}$ is maximum degree, γ is the exponent for the degree distribution, β is the exponent for the community size distribution and μ is the mixing parameter. For a detailed discussion of the parameters the reader is referred to Ref. [137]

to KaHIP, we follow the user guide [2], and use the `kaffpaE` version of the solver with the option `--mh_enable_kabapE` for high quality refinement for perfectly balanced parts. We use the option `--preconfiguration=fast` to ensure results are compared with a single V-cycle. Our results (cut values) are normalized with either the best known value when applicable or by the smallest cut value found by any of the solvers used.

For the Modularity Maximization Problem, we compare our solutions using ML-QLS with two classical clustering methods, Asynchronous Fluid Communities [187] (implemented in NetworkX [101]) and Spectral Clustering [230, 251] (implemented in Scikit-learn [189]). Note that even though these methods solve the same problem (namely, Community Detection or clustering), they do not explicitly maximize modularity. Therefore, it is unfair to directly compare the modularity of the solution produced by them to ML-QLS, which is explicitly maximizing modularity. However, they provide a useful baseline. Moreover, since the maximum possible modularity for at most 2 communities is 0.5, the best solutions found by all methods are no more than 1%–10% away from the optimal (see Table 4.5)

The experimental results are presented in Figure 4.6. We have made all raw result data available on Github [4]. For each problem and method (except for QAOA on IBM Q Poughkeepsie

Network name	Best modularity
finan512-10k	0.499
GirvanNewman	0.459
lancichinetti1	0.452
msc23052	0.499
opsahl-powergrid	0.497
roadNet-PA-20k	0.499

Figure 4.5: Highest modularity value found by all methods for a given problem. The highest possible modularity value for at most 2 communities is 0.5.

quantum computer, labeled "QAOA (IBMQ Poughkeepsie)" in Figure 4.6), we perform ten runs of a single V-cycle with different seeds. For "QAOA (IBMQ Poughkeepsie)", we perform just one run per each problem due to the limited access to quantum hardware.

Observations We observe that ML-QLS is capable of achieving results close to the best ones found by other solvers for all problems. For Graph Partitioning, Figure 4.6 shows significant variability in the quality of the solution across different solvers and problem instances. This effect is also observed for the state-of-the-art Graph Partitioning solver KaHIP, when run for a single V-cycle. This is partially due to the fact that we normalize the objectives to make them directly comparable. For example, for the graph `4elt` the best known cut value presented in The Graph Partitioning Archive [233] is 139. Therefore, an *absolute* difference of 28 edges in cut obtained by a solver translates into a 20% *relative* difference presented in Figure 4.6. However, the same *absolute* difference of 28 edges would translate into $\approx 0.44\%$ for the graph `bcsstk30` (best known cut 6394). The graph `SSS12` is specifically designed to be hard for traditional Graph Partitioning frameworks [215]. This explains the high variation in the performance of KaHIP on it.

It is worth noting that QAOA on the IBM quantum computers (see "QAOA (IBM Q Poughkeepsie)" in Figure 4.6) takes more iterations to converge to a solution compared to D-Wave. This is partially due to the fact that we perform the QAOA variational parameter optimization on the simulator and only run once with the optimized parameters on the device. As a result, the learned variational parameters do not include the noise profile of the device, limiting the quality of subproblem solutions. As devices become more easily available, we expect to be able to run full variational parameter optimization on the quantum hardware.

To project the performance improvements for future hardware, we simulate the performance of ML-QLS as a function of hardware (subproblem) size shown in Figure 4.7. As the subproblem size

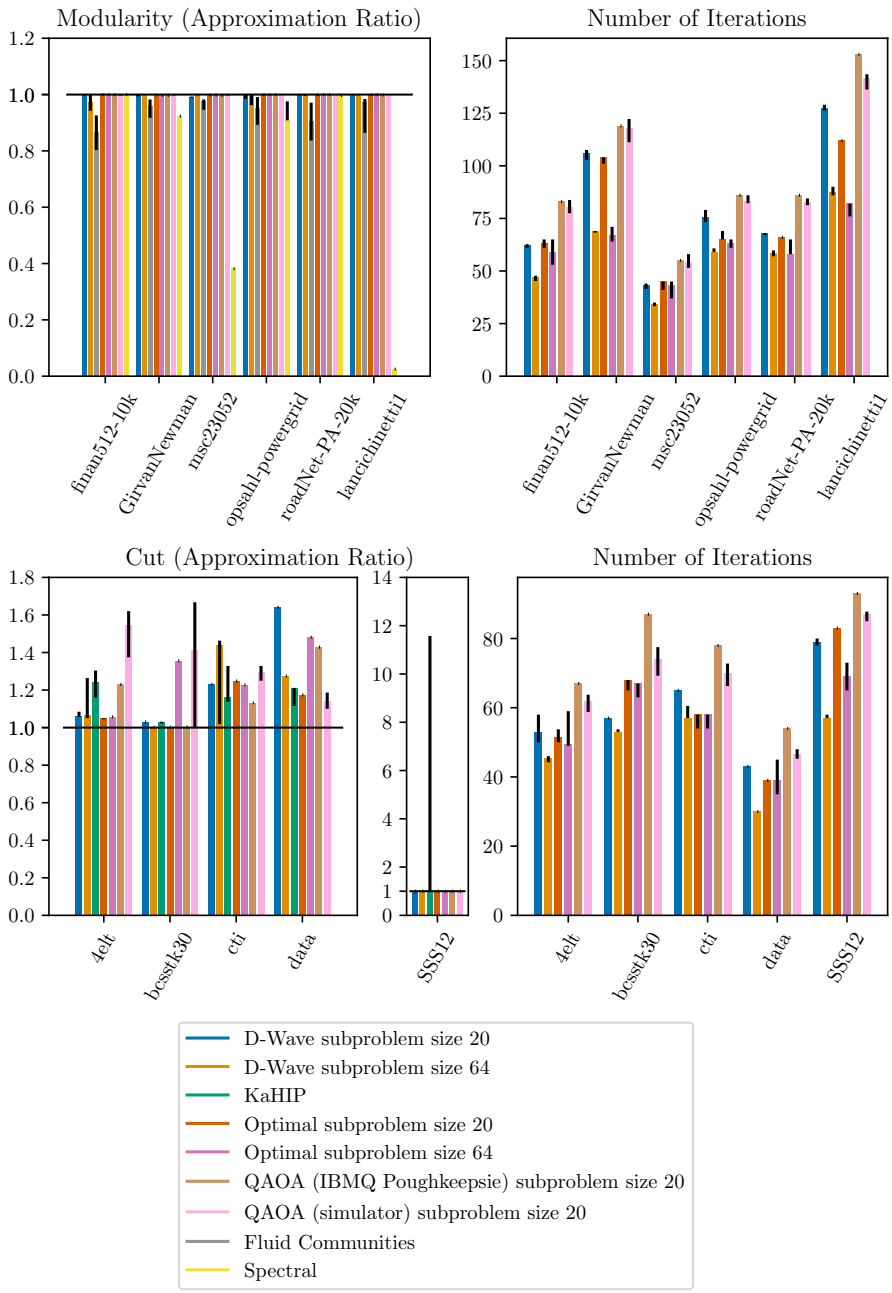
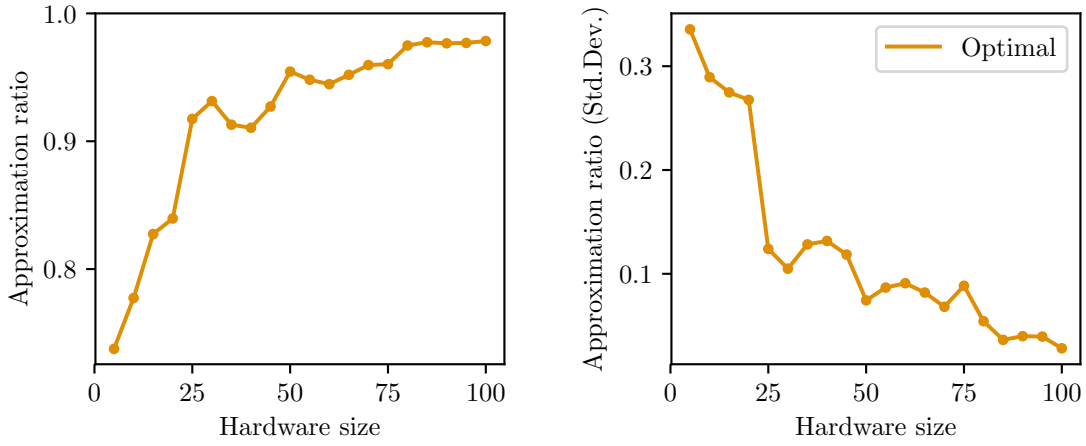


Figure 4.6: Quality of the solution and the number of iterations for all problems and solvers. The height of the bars is the median over 10 seeds. Error bars (black) are 25th and 75th percentiles. For the objective function (Cut or Modularity) all results are normalized by the best solution found by any solver (for Graph Partitioning this includes the best known cuts from The Graph Partitioning Archive [233]). Number of iterations is the number of calls to the subproblem solver (ML-QLS only).



(a) Mean

(b) Standard deviation

Figure 4.7: Modularity (Approximation ratio) as the function of the size of the subproblem (hardware size). The performance is projected using Gurobi as the subproblem solver. The top plot presents the mean approximation ratio averaged over the entire benchmark. The bottom plot presents the standard deviation. As the hardware size increases, the quality of the solution found by ML-QLS improves.

increases, the average quality of the solution found by ML-QLS improves and variation in results decreases. This shows that performance of ML-QLS can be improved as larger size quantum devices and better quantum optimization routines are developed. Note that this makes the assumption that the subproblem is solved to optimality or to a solution that is close to optimal. Evaluating the scaling of quantum optimization algorithms used for solving the subproblems falls outside of the scope of this work (we provide an overview of relevant recent results in Sec. 4.2.1.2).

4.2.3.1 Scaling and running time estimates

The proposed ML-QLS approach is based on the traditional multilevel methods for graph partitioning and graph clustering and therefore a lot of scaling and running time considerations are shared between the two family of methods. Concretely, in our implementation we use the coarsening available in KaHIP graph partitioning package [217], making the running time and the scaling of the *coarsening* stage of our method and KaHIP equal. The running time of solving the problem on the coarsest level does not scale with problem size, as the size of the coarsest level is fixed to be equal to the hardware size. Therefore in this section we will focus on the analysis of the *refinement* stage.

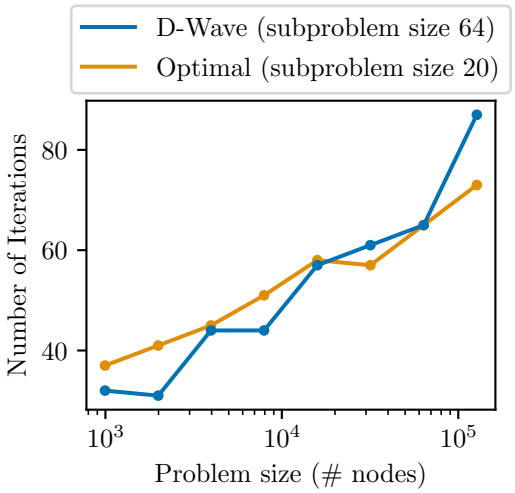


Figure 4.8: The number of iterations (calls to optimizer) to solution for the modularity maximization problem as a function of problem size. The performance is projected using Gurobi as the subproblem solver with subproblem size 20 and allowing it to solve each subproblem to optimality, as well as using D-Wave as the subproblem solver with subproblem size 64. For Gurobi, the subproblem size was limited to 20 to guarantee that each subproblem is proven to be solved the optimum.

To evaluate how the proposed approach scales with the problem size, we construct a series of graphs with the number of nodes ranging from $1,000$ to $128,000$. The graphs are constructed in the same way as the graph `roadNet-PA-20k`, namely by performing a breadth-first search from the median degree node of `roadNet-PA` [136] and including nodes until the desired size is reached. We fix the subproblem size to 20 for Gurobi and to 64 for D-Wave. The results are presented in Fig. 4.8. We note that the number of iterations scales roughly logarithmically with the problem size, as we observe roughly constant number of iterations per uncoarsening level. As we do not constraint the number of refinement iterations, this indicates that the solution projected from coarser levels is of high quality and does not need to be significantly changed at the refinement stage, indicating that the coarsening has successfully constructed a multilevel hierarchy that preserves problem structure. The success of coarsening in this example might be due to the simplicity of the problem structure (planar graph of a road network). However, in all major multilevel solvers the number of refinement calls is artificially limited, preserving the scaling (see Sec. 4.2.1.4 for an in-depth discussion).

The number of iterations to solution presented in Fig. 4.6 and 4.8 allows us to compute rough estimates of the running time of the algorithm. To do that, we need to estimate the running time of QAOA and quantum annealing for our problems. We observe that as the running time

of coarsening is approximately 1 second. for the problems in our benchmark, the overall running time is dominated by refinement. Same is true of this class of multilevel algorithms in general (see Sec. 4.2.1.4).

For QAOA, we are going to follow the assumptions from Ref. [124]. Ref. [124] discusses a cloud platform optimized for variational hybrid algorithms. In this work, we utilized IBM Quantum Experience, which has a different architecture with a queue system, resulting in different latency profile. However, in the idealized scenario of a full access to the machine (i.e. zero wait time in the queue), the projected running times and latencies are very similar to the ones in Ref. [124]. QAOA with hardware-efficient ansatz, as implemented in this work, is a variational algorithm, i.e. to solve the subproblem, the variational parameters have to be optimized (see Sec. 4.2.1.2). In this work, we allowed optimizer 2,000 iterations to find good variational parameters. At each of these steps, the circuit has to be compiled into the native gateset and the topology of the hardware, then converted into pulses and then ran 5,000 times (5,000 "shots") to assemble the statistic on the objective. Assuming the programmer has implemented the circuit in the native gateset and aligned the two-qubit gates with the native topology of the hardware, we can omit the overhead associated with the compilation into the native gateset. For the second step, compilation from gates-level into pulse-level, we will extrapolate the numbers in Ref. [124]. Ref. [124] provides the compilation time of about 50 ms for a two-qubit QAOA circuit with 11 gates. Assuming realistic linear scaling of the translation from gates into pulses, for the 20-qubit RYRZ ansatz used in this work with 99 gates the extrapolated compilation time is 450 ms. Finally, the circuit has to be run for 5,000 shots. For the hardware-efficient ansatz, the circuit depth is fixed and constant (2 two-qubit gates, 4 one-qubit gates in depth). Therefore the per-shot time is dominated by the passive reset time, about 100 μs [124]. Note that the per-shot time varies between hardware implementations. For the device used in Google's quantum supremacy demonstration, the reported time for three million shots is 600 seconds with the limiting time being control hardware communications [17], giving 2 ms per shot. Note that the reported net quantum processor time (30 seconds) gives same 100 μs of net processor time per shot. Running the circuit for 5,000 shots gives estimated runtime of $100\mu\text{s} * 5,000 = 500\text{ms}$ using more optimistic per-shot time estimate. Therefore total running time for one QAOA iteration is equal to $450\text{ms} + 500\text{ms} \approx 1\text{s}$. We used 2,000 iterations to find good QAOA parameters in this work, resulting in 2,000 seconds $\approx$ 30 minutes projected runtime per subproblem. For a problem that takes 60 iterations to solve (e.g. modularity maximization for `4elt`), this gives projected running

time of *about 30 hours*.

This number might give the reader pause. However, we want to point out that numbers in the above estimate can be improved and a number of ways. First, the cost of finding good variational parameters can be reduced drastically by using machine learning techniques. Ref. [133] demonstrates that for an alternating operator ansatz high-quality parameters can be found in as few as 100 iterations. Reducing the number of iterations per subproblem from 2,000 to 100 would reduce the projected running time to 1.5 hours. Further, compiling time can be greatly reduced by compiling the parameterized circuit into pulses once and reusing the compiled pulse schedule [91]. An additional benefit of this approach is that it would allow for better optimization of the pulse schedule, as it can be optimized once offline. Assuming the compilation time can be reduced to that of two-qubit circuit, the projected running time is reduced to ≈ 45 minutes. Finally, the hardware is rapidly evolving and it is not yet clear what architecture will become the standard. Therefore further improvements in running time, while hard to predict, are expected.

For Quantum Annealing on D-Wave, each iteration of the refinement process requires the execution of a single Quantum Machine Instruction (QMI) which includes the QUBO parameters and annealing-cycle parameters sent to the D-Wave system for processing. The total time accessing the QPU (QPU Access time) for a single QMI can be broken down into 4 parts. The first is the one-time initialization step to program the QPU. Then for each sample requested we have the following times. The annealing time, followed by the time needed to wait for the QPU to regain its initial temperature, referred to as the QPU Delay Time, and lastly the time needed to read the sample from the QPU referred to as the QPU Readout Time. In other words,

$$qpu\_access\_time = qpu\_programming\_time + N * (annealing\_time + delay\_time + readout\_time) \tag{4.23}$$

where N is the number of samples requested in a single QMI. The timing $N * (annealing\_time + delay\_time + readout\_time)$ is referred to as the QPU Sampling time. In our experiments we request 1000 samples for each QMI and fixed the annealing time to $20\mu s$ per sample. In order to test and demonstrate scalability of the proposed approach on the D-Wave system, similar to the experiments presented in Figure 4.8, we varied the number of nodes for a road network graph from approximately 10^3 nodes up to 10^5 nodes and computed the maximum modularity of each graph. Similar to the case in Figure 4.8 where each QUBO was solved up to optimality we observe a logarithmic scaling in the

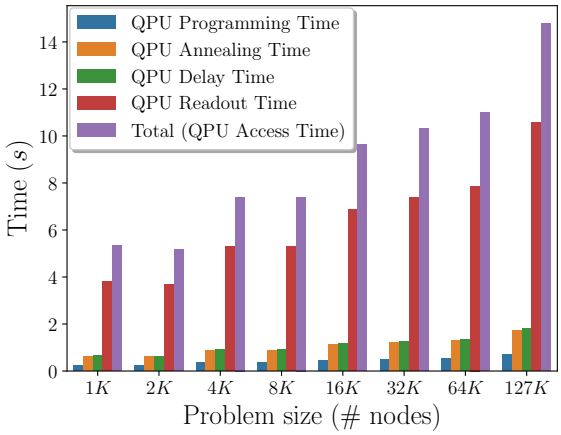


Figure 4.9: D-Wave timing results showing a breakdown of the total time accessing the QPU as the problem size respresented by the number of nodes of the graph increases.

number of calls to the D-Wave system (number of iterations) as shown in Figure 4.8. Figure 4.9 gives a breakdown of the total time (Total QPU Access time) for all iterations to compute the modularity for a given input graph where the smallest graph with approximately 1000 nodes required a QPU Access time of 5 seconds while the largest graph with approximately $127,000$ nodes required a QPU Access time of approximately 15 seconds. However, the majority of the QPU Access time was actually in the QPU Readout time. Requiring less than 2 seconds of total annealing time for 1000 samples is an impressive result for a graph of approximately 127,000 nodes. This time could be significantly reduced by requesting a smaller number of samples per iteration. For example, if one requested a single sample per iteration this would approximately reduce the total QPU sampling time by a factor of 1000. Note that the QPU Access time is a linear function in the number of iterations of the refinement method. Due to the heuristic nature of the proposed approach, the total number of iterations by itself is not a strictly increasing function in the number of nodes of the input graph. Thus for example, we observe in Figure 4.8, the number of iterations is not a strictly increasing function and the graphs with $2K$ and $8K$ nodes required a smaller number of iterations than the graphs with $1K$ and $4K$ nodes respectively. This is subsequently observed in the running times as shown in Figure 4.9.

4.2.4 Open Problems and Discussion

Revising (un)coarsening operators in anticipation of the new class of high-quality refinement solvers is the first major open problem. The majority of multilevel algorithms for combinatorial optimization problems are inspired by the idea of "thinking globally while acting locally". However, there is a crucial difference between these algorithms for combinatorial problems and such methods as multigrid for continuous problems or multiscale PDE-based optimization. In multigrid (e.g., for systems of linear equations), a relaxation at the uncoarsening stage is convergent [38], and in most cases assumes an optimal solution (up to some tolerance) for a subset of relaxed variables given other variables are invariant (i.e., a fixed solution for those variables that are not in the optimized subset). Examples include easily parallelizable Jacobi relaxation, as well as hard to parallelize Gauss-Seidel relaxation in which most variables are typically optimized sequentially, and many more. Both coarsening and uncoarsening operators (also known as the restriction, and prolongation in multigrid) assume this convergence which in the end provides guarantees for the entire multilevel framework. However, for the *combinatorial* multilevel solvers, the integer variables make this assumption practically impossible, even for subproblems containing tens of variables optimized simultaneously. With the development of less noisy quantum devices, we can assume that in our hands will be extremely fast heuristics to produce nearly (if hypothetically not completely) optimal solutions for combinatorial optimization problems of up to several hundreds variables. In order to use the multilevel paradigm correctly, there will be a critical need to revise (un)coarsening operators that take this feature into account because (to the best of our knowledge) all existing versions of coarsening operators do not consider optimality of the refinement. Moreover, most existing multilevel frameworks exhibit more emphasis on computational speedup rather than on the quality of the solution to better approximate the fine problem.

The second problem is not unique to multilevel methods but to most decomposition based approaches. Even if quantum devices become fully developed and become more accessible for the broad scientific community, they will still remain more expensive than regular CPU based devices. The decomposition approaches split the problem into many small local subproblems, while multilevel methods may need even more of them because solving subproblems is required at all levels of coarseness. Thus, there is a critical need in developing an extremely fast routing classifier for a subproblem that will decide whether solving a particular subproblem on the NISQ device will be

beneficial in comparison to the CPU.

4.2.5 Conclusion

Current Noisy Intermediate-Scale Quantum (NISQ) devices are limited in the number of qubits and can therefore only be used to directly solve combinatorial optimization problems that exhibit a limited number of variables. In order to overcome this limitation, in this work we have proposed the multilevel computational framework for solving large-scale combinatorial problems on NISQ devices. We demonstrate this approach on two well-known combinatorial optimization problems, the Graph Partitioning Problem, and the Community Detection Problem, and perform experiments on the 20 qubit IBM gate-model quantum computer, and the 2048 qubit D-Wave 2000Q quantum annealer. In order to implement an efficient iterative refinement scheme using the NISQ devices, we have developed novel techniques for efficiently formulating and evaluating sub-QUBOs without explicitly constructing the entire QUBO of the large-scale problem, which in many cases can be a dense matrix that makes it computationally expensive to store and process. In our experiments, for the Graph Partitioning Problem, five graphs were chosen such that the smallest graph had 2851 nodes while the largest had 28924 nodes, while for the Community Detection Problem, the smallest graph had 4941 nodes and largest had 10,000 nodes. For both problems, for comparison purposes, we run one V-cycle of the multilevel framework with the different NISQ devices multiple times and compared the results to the state-of-art methods. Our experimental results give comparable results to the state-of-the-art methods and for some cases we were able to get the best-known results. This work therefore provides an important stepping stone to demonstrating practical Quantum Advantage. As the capabilities of NISQ devices increase, we are hopeful that similar methods can provide a path to adoption of quantum computers for a variety of business and scientific applications.

Bibliography

[1] URL https://github.com/rsln-s/Subspace-symmetry-predicts-QAOA-performance.

[2] KaHIP v2.10 – Karlsruhe High Quality Partitioning User Guide. http://algo2.iti.kit.edu/schulz/software_releases/kahipv2.10.pdf. [Online; accessed July 31, 2019].

[3] . URL https://github.com/rsln-s/ml_qls.

[4] . URL https://github.com/rsln-s/ml_qls/tree/bc376276ba684460aeccaa371b4fc38003139e34/multilevel/data/results_csv.

[5] URL https://www.dropbox.com/s/cftspvdoovnzi4l/allresults.p.zip?dl=0.

[6] Emile Aarts and Jan K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997. ISBN 0471948225.

[7] Antonio Acín, Immanuel Bloch, Harry Buhrman, Tommaso Calarco, Christopher Eichler, Jens Eisert, Daniel Esteve, Nicolas Gisin, Steffen J Glaser, Fedor Jelezko, et al. The quantum technologies roadmap: a european community view. *New Journal of Physics*, 20(8):080201, 2018.

[8] Ravindra K Ahuja, Özlem Ergun, James B Orlin, and Abraham P Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, 123(1-3):75–102, 2002.

[9] Yaroslav Akhremtsev, Tobias Heuer, Peter Sanders, and Sebastian Schlag. Engineering a direct k-way hypergraph partitioning algorithm. In *2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 28–42. SIAM, 2017.

[10] Tameem Albash and Daniel A. Lidar. Demonstration of a scaling advantage for a quantum annealer over simulated annealing. *Physical Review X*, 8(3), July 2018. doi: 10.1103/physrevx.8.031016. URL https://doi.org/10.1103/physrevx.8.031016.

[11] Tameem Albash and Daniel A Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018.

[12] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernádez, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, Jerry M. Chow, Antonio D. Córcoles-Gonzales, Abigail J. Cross, Andrew Cross, Juan Cruz-Benito, Chris Culver, Salvador De La Puente González, Enrique De La Torre, Delton Ding, Eugene Dumitrescu, Ivan Duran, Pieter Eendebak, Mark Everitt, Ismael Faro Sertage, Albert Frisch, Andreas Fuhrer, Jay Gambetta, Borja Godoy Gago, Juan Gomez-Mosquera, Donny Greenberg, Ikko Hamamura, Vojtech Havlicek, Joe Hellmers, Łukasz Herok, Hiroshi Horii, Shaohan Hu, Takashi Imamichi, Toshinari Itoko, Ali Javadi-Abhari,

Naoki Kanazawa, Anton Karazeev, Kevin Krsulich, Peng Liu, Yang Luh, Yunho Maeng, Manoel Marques, Francisco Jose Martín-Fernández, Douglas T. McClure, David McKay, Srujan Meesala, Antonio Mezzacapo, Nikolaj Moll, Diego Moreda Rodríguez, Giacomo Nannicini, Paul Nation, Pauline Ollitrault, Lee James O'Riordan, Hanhee Paik, Jesús Pérez, Anna Phan, Marco Pistoia, Viktor Prutyanov, Max Reuter, Julia Rice, Abdón Rodríguez Davila, Raymond Harry Putra Rudy, Mingi Ryu, Ninad Sathaye, Chris Schnabel, Eddie Schoute, Kanav Setia, Yunong Shi, Adenilton Silva, Yukio Siraichi, Seyon Sivarajah, John A. Smolin, Mathias Soeken, Hitomi Takahashi, Ivano Tavernelli, Charles Taylor, Pete Taylour, Kenso Trabing, Matthew Treinish, Wes Turner, Desiree Vogt-Lee, Christophe Vuillot, Jonathan A. Wildstrom, Jessica Wilson, Erick Winston, Christopher Wood, Stephen Wood, Stefan Wörner, Ismail Yunus Akhalwaya, and Christa Zoufal. Qiskit: An open-source framework for quantum computing, 2019.

[13] Charles J Alpert and Andrew B Kahng. Recent directions in netlist partitioning: a survey. *Integration, the VLSI journal*, 19(1-2):1–81, 1995.

[14] Andris Ambainis, Kaspars Balodis, Jānis Iraids, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. Quantum speedups for exponential-time dynamic programming algorithms. *arXiv preprint arXiv:1807.05209*, 2018.

[15] Eric Anschuetz, Jonathan Olson, Alán Aspuru-Guzik, and Yudong Cao. Variational quantum factoring. In *International Workshop on Quantum Technology and Optimization Problems*, pages 74–85. Springer, 2019.

[16] Sanjeev Arora, David Karger, and Marek Karpinski. Polynomial time approximation schemes for dense instances of NP-hard problems. *Journal of Computer and System Sciences*, 58(1): 193–210, February 1999. doi: 10.1006/jcss.1998.1605. URL `https://doi.org/10.1006/jcss.1998.1605`.

[17] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, David A. Buell, Brian Burkett, Yu Chen, Zijun Chen, Ben Chiaro, Roberto Collins, William Courtney, Andrew Dunsworth, Edward Farhi, Brooks Foxen, Austin Fowler, Craig Gidney, Marissa Giustina, Rob Graff, Keith Guerin, Steve Habegger, Matthew P. Harrigan, Michael J. Hartmann, Alan Ho, Markus Hoffmann, Trent Huang, Travis S. Humble, Sergei V. Isakov, Evan Jeffrey, Zhang Jiang, Dvir Kafri, Kostyantyn Kechedzhi, Julian Kelly, Paul V. Klimov, Sergey Knysh, Alexander Korotkov, Fedor Kostritsa, David Landhuis, Mike Lindmark, Erik Lucero, Dmitry Lyakh, Salvatore Mandrà, Jarrod R. McClean, Matthew McEwen, Anthony Megrant, Xiao Mi, Kristel Michielsen, Masoud Mohseni, Josh Mutus, Ofer Naaman, Matthew Neeley, Charles Neill, Murphy Yuezhen Niu, Eric Ostby, Andre Petukhov, John C. Platt, Chris Quintana, Eleanor G. Rieffel, Pedram Roushan, Nicholas C. Rubin, Daniel Sank, Kevin J. Satzinger, Vadim Smelyanskiy, Kevin J. Sung, Matthew D. Trevithick, Amit Vainsencher, Benjamin Villalonga, Theodore White, Z. Jamie Yao, Ping Yeh, Adam Zalcman, Hartmut Neven, and John M. Martinis. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, October 2019. doi: 10.1038/s41586-019-1666-5. URL `https://doi.org/10.1038/s41586-019-1666-5`.

[18] Ryan Babbush, Nathan Wiebe, Jarrod McClean, James McClain, Hartmut Neven, and Garnet Kin-Lic Chan. Low-depth quantum simulation of materials. *Physical Review X*, 8(1), March 2018. doi: 10.1103/physrevx.8.011044. URL `https://doi.org/10.1103/physrevx.8.011044`.

[19] Fabian Ball and Andreas Geyer-Schulz. How symmetric are real-world graphs? A large-scale study. *Symmetry*, 10(1):29, Jan 2018. ISSN 2073-8994. doi: 10.3390/sym10010029. URL `http://dx.doi.org/10.3390/sym10010029`.

[20] CJ Ballance, TP Harty, NM Linke, MA Sepiol, and DM Lucas. High-fidelity quantum logic gates using trapped-ion hyperfine qubits. *Physical Review Letters*, 117(6):060504, 2016.

[21] Boaz Barak, Ankur Moitra, Ryan O'Donnell, Prasad Raghavendra, Oded Regev, David Steurer, Luca Trevisan, Aravindan Vijayaraghavan, David Witmer, and John Wright. Beating the random assignment on constraint satisfaction problems of bounded degree. *arXiv:1505.03424*, 2015.

[22] Giampiero Bardella, Angelo Bifone, Andrea Gabrielli, Alessandro Gozzi, and Tiziano Squartini. Hierarchical organization of functional connectivity in the mouse brain: A complex network approach. *Scientific Reports*, 6:32060, 2016.

[23] Rami Barends, Julian Kelly, Anthony Megrant, Andrzej Veitia, Daniel Sank, Evan Jeffrey, Ted C White, Josh Mutus, Austin G Fowler, Brooks Campbell, et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500, 2014.

[24] Stephen T Barnard and Horst D Simon. Fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency and computation: Practice and Experience*, 6(2):101–117, 1994.

[25] Stefanie Barz, Elham Kashefi, Anne Broadbent, Joseph F Fitzsimons, Anton Zeilinger, and Philip Walther. Demonstration of blind quantum computing. *science*, 335(6066):303–308, 2012.

[26] Vladimir Batagelj and Ulrik Brandes. Efficient generation of large random networks. *Physical Review E*, 71(3):036113, 2005.

[27] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, and Nathan Killoran. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv:1811.04968*, 2018.

[28] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[29] Zhengbing Bian, Fabian Chudak, Robert Israel, Brad Lackey, William G Macready, and Aidan Roy. Discrete optimization using quantum annealing on sparse ising models. *Frontiers in Physics*, 2:56, 2014.

[30] Zhengbing Bian, Fabian Chudak, Robert Brian Israel, Brad Lackey, William G Macready, and Aidan Roy. Mapping constrained optimization problems to quantum annealing with application to fault diagnosis. *Frontiers in ICT*, 3:14, 2016.

[31] Charles-Edmond Bichot and Patrick Siarry. *Graph partitioning.* John Wiley & Sons, 2013.

[32] Norman Biggs. *Algebraic graph theory*, volume 67. Cambridge university press, 1993.

[33] Marianna Bolla. Spectra, euclidean representations and clusterings of hypergraphs. *Discrete Mathematics*, 117(1-3):19–39, 1993.

[34] Michael Booth, Steven P Reinhardt, and Aidan Roy. Partitioning optimization problems for hybrid classical. *quantum execution. Technical Report*, pages 01–09, 2017.

[35] Tomas Boothby, Andrew D King, and Aidan Roy. Fast clique minor generation in chimera qubit connectivity graphs. *Quantum Information Processing*, 15(1):495–508, 2016.

[36] Fernando GSL Brandao, Michael Broughton, Edward Farhi, Sam Gutmann, and Hartmut Neven. For fixed control parameters the quantum approximate optimization algorithm's objective function value concentrates for typical instances. *arXiv:1812.04170*, 2018.

[37] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. *arXiv:physics/0608255*, 2006.

[38] A. Brandt. Multiscale Scientific Computation: Review 2001. In *Multiscale and Multiresolution Methods*, volume 20 of *LNCSE*, pages 3–95. Springer, 2002. ISBN 978-3-540-42420-8. URL `http://dx.doi.org/10.1007/978-3-642-56205-1_1`.

[39] A. Brandt and D. Ron. Chapter 1 : Multigrid solvers and multilevel optimization strategies. In J. Cong and J. R. Shinnerl, editors, *Multilevel Optimization and VLSICAD*. Kluwer, 2003.

[40] Sergey Bravyi, Graeme Smith, and John A. Smolin. Trading classical and quantum computational resources. *Physical Review X*, 6(2):021043, 2016.

[41] Sergey Bravyi, Alexander Kliesch, Robert Koenig, and Eugene Tang. Obstacles to state preparation and variational optimization from symmetry protection. *arXiv:1910.08980*, 2019.

[42] Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[43] William L Briggs, Van Emden Henson, and Steve F McCormick. *A multigrid tutorial*. SIAM, 2000.

[44] Aydin Buluc and Erik G. Boman. Towards scalable parallel hypergraph partitioning. In *CSRI Summer Proceedings 2008*, pages 109–119. Sandia National Labs, 2008.

[45] Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. Recent advances in graph partitioning. In *Algorithm Engineering*, pages 117–158. Springer, 2016.

[46] Ümit Çatalyürek and Cevdet Aykanat. Patoh (partitioning tool for hypergraphs). In *Encyclopedia of Parallel Computing*, pages 1479–1487. Springer, 2011.

[47] Umit V Catalyurek and Cevdet Aykanat. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Trans. Parallel Distrib. Syst.*, 10(7):673–693, 1999.

[48] Umit V Catalyurek, Erik G Boman, Karen D Devine, Doruk Bozdağ, Robert T Heaphy, and Lee Ann Riesen. A repartitioning hypergraph model for dynamic load balancing. *J. Parallel Distrib. Comput.*, 69(8):711–724, 2009.

[49] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J. Coles. Cost-function-dependent barren plateaus in shallow quantum neural networks, 2020.

[50] TH Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. Spectral properties of hypergraph laplacian and approximation algorithms. *arXiv preprint arXiv:1605.01483*, 2016.

[51] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. acm transactions on intelligent systems and technology, 2: 27: 1–27: 27, 2011. *Software available at http://www. csie. ntu. edu. tw/cjlin/libsvm*, 2011.

[52] Jie Chen and Ilya Safro. Algebraic distance on graphs. *SIAM J. Sci. Comput.*, 33(6):3468–3490, 2011.

[53] Jingchun Chen and Bo Yuan. Detecting functional modules in the yeast protein–protein interaction network. *Bioinformatics*, 22(18):2283–2290, 2006.

[54] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. American Mathematical Soc., 1997.

[55] J. Cong and J. R. Shinnerl, editors. *Multilevel Optimization and VLSICAD*. Kluwer, 2003. ISBN 1-4020-1081-8.

[56] International Business Machines Corporation. IBM ILOG CPLEX v 12.9.0, . URL `https://www.ibm.com/products/ilog-cplex-optimization-studio`.

[57] International Business Machines Corporation. IBM QISKit Aqua: Variatinal forms. `https://github.com/Qiskit/aqua/blob/master/qiskit_aqua/algorithms/components/variational_forms/ryrz.py`, . [Online; accessed September 21, 2018].

[58] International Business Machines Corporation. IBM Q: Quantum devices and simulators. `https://www.research.ibm.com/ibm-q/technology/devices/`, . [Online; accessed September 21, 2018].

[59] Gavin E Crooks. Performance of the quantum approximate optimization algorithm on the maximum cut problem. *arXiv:1811.08419*, 2018.

[60] Andrew Cross. The ibm q experience and qiskit open-source quantum computing software. *Bulletin of the American Physical Society*, 2018.

[61] D-Wave Systems Inc. Introduction to the D-Wave quantum hardware, 2018. URL `www.dwavesys.com/tutorials/background-reading-series/introduction-d-wave-quantum-hardware`.

[62] D-Wave Systems Inc. Measuring computation time on d-wave systems. *D-Wave User Manual 09-1107A-M*, 2019. URL `https://docs.dwavesys.com/docs/latest/doc_timing.html`.

[63] D-Wave Systems, Inc. D-Wave system documentation: Problem decomposition, 2019. URL `https://docs.dwavesys.com/docs/latest/c_handbook_7.html#toolsdecomposition`. [Online; accessed 28-March-2019].

[64] Timothy A Davis and Yifan Hu. The university of florida sparse matrix collection. *ACM Trans. Math. Software*, 38(1):1, 2011.

[65] Timothy A Davis, William W Hager, Scott P Kolodziej, and S Nuri Yeralan. Algorithm xxx: Mongoose, a graph coarsening and partitioning library. *ACM Trans. Math. Software*, 2019.

[66] David Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proc. R. Soc. Lond. A*, 400(1818):97–117, 1985.

[67] Karen D Devine, Erik G Boman, Robert T Heaphy, Rob H Bisseling, and Umit V Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 10–pp. IEEE, 2006.

[68] Karen D Devine, Vitus Leung, Erik G Boman, Sivasankaran Rajamanickam, Lee Ann Riesen, and Umit Catalyurek. Zoltan user's guide, version 3.8.(2014), 2014. URL `http://www.cs.sandia.gov/zoltan/ug_html/ug_alg_patoh.html`.

[69] Harris Drucker, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. Support vector regression machines. In *Advances in neural information processing systems*, pages 155–161, 1997.

[70] Vedran Dunjko, Yimin Ge, and J Ignacio Cirac. Computational speedups using small quantum devices. *arXiv preprint arXiv:1807.08970*, 2018.

[71] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[72] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–475, April 2001. doi: 10.1126/science.1057726. URL `https://doi.org/10.1126/science.1057726`.

[73] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv:1602.07674*, 2016.

[74] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[75] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv:1411.4028*, 2014.

[76] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. *arXiv:1412.6062*, 2014.

[77] Edward Farhi, Jeffrey Goldkanstone, Sam Gutmann, and Hartmut Neven. Quantum algorithms for fixed qubit architectures. *arXiv:1703.06199*, 2017.

[78] Edward Farhi, David Gamarnik, and Sam Gutmann. The quantum approximate optimization algorithm needs to see the whole graph: A typical case, 2020.

[79] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*, pages 241–247. ACM, 1988.

[80] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 23(2): 298–305, 1973.

[81] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[82] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[83] Artur Garcia-Saez and Jordi Riu. Quantum observables for continuous control of the quantum approximate optimization algorithm via reinforcement learning. *arXiv preprint arXiv:1911.09682*, 2019.

[84] Michael R Gary and David S Johnson. Computers and intractability: A guide to the theory of np-completeness, 1979.

[85] Mirco Gelain, Maria Silvia Pini, Francesca Rossi, K Brent Venable, and Toby Walsh. Local search approaches in stable matching problems. *Algorithms*, 6(4):591–617, 2013.

[86] E Gelman and J Mandel. On multilevel iterative methods for optimization problems. *Mathematical Programming*, 48(1-3):1–17, 1990.

[87] Giorgos Georgiadis and Marina Papatriantafilou. Overlays with preferences: Distributed, adaptive approximation algorithms for matching with preference lists. *Algorithms*, 6(4):824–856, 2013.

[88] Vittorio Giovannetti, Lorenzo Maccone, Tomoyuki Morimae, and Terry G Rudolph. Efficient universal blind quantum computation. *Physical review letters*, 111(23):230501, 2013.

[89] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, June 2002. doi: 10.1073/pnas.122653799. URL https://doi.org/10.1073/pnas.122653799.

[90] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42 (6):1115–1145, 1995.

[91] Pranav Gokhale, Yongshan Ding, Thomas Propson, Christopher Winkler, Nelson Leung, Yunong Shi, David I. Schuster, Henry Hoffmann, and Frederic T. Chong. Partial compilation of variational algorithms for noisy intermediate-scale quantum machines. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, October 2019. doi: 10.1145/3352460.3358313. URL https://doi.org/10.1145/3352460.3358313.

[92] Serge Gratton, Annick Sartenaer, and Philippe L Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM Journal on Optimization*, 19(1):414–444, 2008.

[93] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219. ACM, 1996.

[94] Emily Grumbling and Mark Horowitz. Quantum computing: Progress and prospects. *National Academies of Sciences, Engineering, and Medicine*, 2018. doi: 10.17226/25196.

[95] G. G. Guerreschi and A. Y. Matsuura. QAOA for max-cut requires hundreds of qubits for quantum speed-up. *Scientific Reports*, 9(1), May 2019. doi: 10.1038/s41598-019-43176-9. URL https://doi.org/10.1038/s41598-019-43176-9.

[96] Gian Giacomo Guerreschi and Mikhail Smelyanskiy. Practical optimization for hybrid quantum-classical algorithms. *arXiv:1701.01450*, 2017.

[97] Dan Gusfield and Robert W Irving. *The stable marriage problem: structure and algorithms*. MIT press, 1989.

[98] Stuart Hadfield. Quantum algorithms for scientific computing and approximate optimization. *Columbia university PhD dissertation, arXiv:1805.03265*, 2018.

[99] Stuart Hadfield. On the representation of Boolean and real functions as Hamiltonians for quantum computing. *arXiv:1804.09130*, 2018.

[100] Stuart Hadfield, Zhihui Wang, Bryan O'Gorman, Eleanor G Rieffel, Davide Venturelli, and Rupak Biswas. From the quantum approximate optimization algorithm to a quantum alternating operator ansatz. *arXiv preprint arXiv:1709.03489*, 2017.

[101] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, pages 11–15, Pasadena, CA USA, 2008.

[102] Lars Hagen and Andrew B Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 11(9):1074–1085, 1992.

[103] William W Hager, James T Hungerford, and Ilya Safro. A multilevel bilinear programming algorithm for the vertex separator problem. *Computational Optimization and Applications*, 69 (1):189–223, 2018.

[104] Ryan Hamerly, Takahiro Inagaki, Peter L. McMahon, Davide Venturelli, Alireza Marandi, Tatsuhiro Onodera, Edwin Ng, Carsten Langrock, Kensuke Inaba, Toshimori Honjo, Koji Enbutsu, Takeshi Umeki, Ryoichi Kasahara, Shoko Utsunomiya, Satishi Kako, Ken-ichi Kawarabayashi, Robert L Byer, Martin M Fejer, Hideo Mabuchi, Dirk Englund, Eleanor Rieffel, Hiroki Takesue, and Yoshihisa Yamamoto. Experimental investigation of performance differences between coherent ising machines and a quantum annealer. *Science Advances*, 5(5): eaau0823, 2019.

[105] Niels W Hanson, Kishori M Konwar, Alyse K Hawley, Tomer Altman, Peter D Karp, and Steven J Hallam. Metabolic pathways for the whole community. *BMC genomics*, 15(1):619, 2014.

[106] R. Harris, Y. Sato, A. J. Berkley, M. Reis, F. Altomare, M. H. Amin, K. Boothby, P. Bunyk, C. Deng, C. Enderud, S. Huang, E. Hoskinson, M. W. Johnson, E. Ladizinsky, N. Ladizinsky, T. Lanting, R. Li, T. Medina, R. Molav, R. Neufeld, T. Oh, I. Pavlov, I. Perminov, G. Poulin-Lamarre, C. Rich, A. Smirnov, L. Swenson, N. Tsai, M. Volkmann, J. Whittaker, and J. Yao. Phase transitions in a programmable quantum spin glass simulator. *Science*, 361:162–165, 2017.

[107] Aram Harrow and John Napp. Low-depth gradient measurements can improve convergence in variational hybrid quantum-classical algorithms. *arXiv:1901.05374*, 2019.

[108] William E Hart, Carl D Laird, Jean-Paul Watson, David L Woodruff, Gabriel A Hackebeil, Bethany L Nicholson, and John D Siirola. *Pyomo-optimization modeling in python*, volume 67. Springer, 2017.

[109] Matthew B. Hastings. Classical and quantum bounded depth approximation algorithms. *arXiv:1905.07047*, 2019.

[110] Bruce Hendrickson and Robert W Leland. A multi-level algorithm for partitioning graphs. *SC*, 95(28), 1995.

[111] Michael A Heroux, Roscoe A Bartlett, Vicki E Howle, Robert J Hoekstra, Jonathan J Hu, Tamara G Kolda, Richard B Lehoucq, Kevin R Long, Roger P Pawlowski, Eric T Phipps, et al. An overview of the trilinos project. *ACM Trans. Math. Software*, 31(3):397–423, 2005.

[112] Manuel Holtgrewe, Peter Sanders, and Christian Schulz. Engineering a scalable high quality graph partitioner. In *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*. IEEE, 2010. doi: 10.1109/ipdps.2010.5470485. URL `https://doi.org/10.1109/ipdps.2010.5470485`.

[113] Reiner Horst, Panos M Pardalos, and Nguyen Van Thoai. *Introduction to global optimization*. Springer Science & Business Media, 2000.

[114] Shenglong Hu and Liqun Qi. Algebraic connectivity of an even uniform hypergraph. *J. Comb. Optim.*, 24(4):564–579, 2012.

[115] Cupjin Huang, Mario Szegedy, Fang Zhang, Xun Gao, Jianxin Chen, and Yaoyun Shi. Alibaba cloud quantum development platform: Applications to quantum algorithm design, 2019.

[116] Stephen Hudson, Jeffrey Larson, Stefan M. Wild, and David Bindel. libEnsemble users manual, 2019. URL `https://buildmedia.readthedocs.org/media/pdf/libensemble/latest/libensemble.pdf`.

[117] IARPA. Quantum Enhanced Optimization (QEO). `https://www.iarpa.gov/index.php/research-programs/qeo`. [Online; accessed September 25, 2018].

[118] Edmund Ihler, Dorothea Wagner, and Frank Wagner. Modeling hypergraphs by graphs with the same mincut properties. *Inf. Process. Lett.*, 45(4):171–175, 1993.

[119] Zhang Jiang, Eleanor G Rieffel, and Zhihui Wang. Near-optimal quantum circuit for Grover's unstructured search using a transverse field. *Physical Review A*, 95(6):062317, 2017.

[120] DS Johnson. Local search and the traveling salesman problem. In *Proceedings of 17th International Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science,(Springer-Verlag, Berlin, 1990)*, pages 443–460, 1990.

[121] Steven G Johnson. The NLopt nonlinear-optimization package, 2019. URL `http://github.com/stevengj/nlopt`.

[122] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed September 21, 2018].

[123] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242, 2017.

[124] Peter J Karalekas, Nikolas A Tezak, Eric C Peterson, Colm A Ryan, Marcus P da Silva, and Robert S Smith. A quantum-classical cloud platform optimized for variational hybrid algorithms. *Quantum Science and Technology*, 5(2):024003, April 2020. doi: 10.1088/2058-9565/ab7559. URL `https://doi.org/10.1088/2058-9565/ab7559`.

[125] Hamed Karimi and Gili Rosenberg. Boosting quantum annealer performance via sample persistence. *Quantum Information Processing*, 16(7):166, 2017.

[126] Hamed Karimi, Gili Rosenberg, and Helmut G Katzgraber. Effective optimization using sample persistence: A case study on quantum annealers and various monte carlo optimization methods. *Physical Review E*, 96(4):043312, 2017.

[127] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Springer, 1972.

[128] G. Karypis and V. Kumar. Analysis of multilevel graph partitioning. Technical Report TR-95-037, Computer Science Dept., Univ. of Minnesota, Minneapolis, MN, 1995.

[129] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1998.

[130] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. Multilevel hypergraph partitioning: applications in vlsi domain. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 7(1):69–79, 1999.

[131] Carl T Kelley. *Iterative methods for optimization*. SIAM, 1999.

[132] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, 49(2):291–307, 1970.

[133] Sami Khairy, Ruslan Shaydulin, Lukasz Cincio, Yuri Alexeev, and Prasanna Balaprakash. Learning to optimize variational quantum circuits to solve combinatorial problems. *Proceedings of the Thirty-Forth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2019.

[134] Andrew D King, Juan Carrasquilla, Jack Raymond, Isil Ozfidan, Evgeny Andriyash, Andrew Berkley, Mauricio Reis, Trevor Lanting, Richard Harris, Fabio Altomare, et al. Observation of topological phenomena in a programmable lattice of 1,800 qubits. *Nature*, 560(7719):456, 2018.

[135] Andrew D King, Jack Raymond, Trevor Lanting, Sergei V Isakov, Masoud Mohseni, Gabriel Poulin-Lamarre, Sara Ejtemaee, William Bernoudy, Isil Ozfidan, Anatoly Y Smirnov, Mauricio Reis, Fabio Altomare, Michael Babcock, Catia Baron, Andrew J Berkley, Kelly Boothby, Paul I Bunyk, Holly Christiani, Colin Enderud, Bram Evert, Richard Harris, Emile Hoskinson, Shuiyuan Huang, Kais Jooya, Ali Khodabandelou, Nicolas Ladizinsky, Ryan Li, P Aaron Lott, Allison J R MacDonald, Danica Marsden, Gaelen Marsden, Teresa Medina, Reza Molavi, Richard Neufeld, Mana Norouzpour, Travis Oh, Igor Pavlov, Ilya Perminov, Thomas Prescott, Chris Rich, Yuki Sato, Benjamin Sheldan, George Sterling, Loren J Swenson, Nicholas Tsai, Mark H Volkmann, Jed D. Whittaker, Warren Wilkinson, Jason Yao, Hartmut Neven, Jeremy P Hilton, Eric Ladizinsky, Mark W Johnson, and Mohammad H Amin. Scaling advantage in quantum simulation of geometrically frustrated magnets. *arXiv preprint arXiv:1911.03446*, 2019.

[136] Jérôme Kunegis. Konect: the koblenz network collection. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1343–1350. ACM, 2013.

[137] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4), October 2008. doi: 10.1103/physreve.78.046110. URL `https://doi.org/10.1103/physreve.78.046110`.

[138] Jeffrey Larson and Stefan M. Wild. A batch, derivative-free algorithm for finding multiple local minima. *Optimization and Engineering*, 17(1):205–228, 2016. doi: 10.1007/s11081-015-9289-7.

[139] Jeffrey Larson and Stefan M. Wild. Asynchronously parallel optimization solver for finding multiple minima. *Mathematical Programming Computation*, 10(3):303–332, 2018. doi: 10.1007/s12532-017-0131-4.

[140] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[141] Sven Leyffer and Ilya Safro. Fast response to infection spread and cyber attacks on large-scale networks. *Journal of Complex Networks*, 1(2):183–199, 2013.

[142] W Liu, D Wilkins, and B Alidaee. A hybrid multi-exchange local search for unconstrained binary quadratic program. *University of Mississippi, Hearin Center for Enterprise Science, HCES-09-05*, 2005.

[143] Oren E Livne and Achi Brandt. Lean algebraic multigrid (lamg): Fast graph laplacian linear solver. *SIAM Journal on Scientific Computing*, 34(4):B499–B522, 2012.

[144] Gui Lu Long and Li Xiao. Experimental realization of a fetching algorithm in a 7-qubit nmr spin liouville space computer. *The Journal of Chemical Physics*, 119(16):8473–8481, 2003.

[145] Gui Lu Long and Li Xiao. Parallel quantum computing in a single ensemble quantum computer. *Physical Review A*, 69(5):052303, 2004.

[146] Foad Lotfifar and Matthew Johnson. A multi–level hypergraph partitioning algorithm using rough set clustering. In *European Conference on Parallel Processing*, pages 159–170. Springer, 2015.

[147] Enyue Lu and SQ Zheng. A parallel iterative improvement stable matching algorithm. In *International Conference on High-Performance Computing*, pages 55–65. Springer, 2003.

[148] Ben D MacArthur, Rubén J Sánchez-García, and James W Anderson. Symmetry in complex networks. *Discrete Applied Mathematics*, 156(18):3525–3531, 2008.

[149] Salvatore Mandrà, Zheng Zhu, Wenlong Wang, Alejandro Perdomo-Ortiz, and Helmut G. Katzgraber. Strengths and weaknesses of weak-strong cluster problems: A detailed overview of state-of-the-art classical heuristics versus quantum approaches. *Phys. Rev. A*, 94:022337, Aug 2016. doi: 10.1103/PhysRevA.94.022337. URL https://link.aps.org/doi/10.1103/PhysRevA.94.022337.

[150] Salvatore Mandrà, Helmut G Katzgraber, and Creighton Thomas. The pitfalls of planar spin-glass benchmarks: raising the bar for quantum annealers (again). *Quantum Science and Technology*, 2(3):038501, jul 2017. doi: 10.1088/2058-9565/aa7877. URL https://doi.org/10.1088%2F2058-9565%2Faa7877.

[151] Fredrik Manne, Md. Naim, Håkon Lerring, and Mahantesh Halappanavar. *On Stable Marriages and Greedy Matchings*, pages 92–101. SIAM, 2016. doi: 10.1137/1.9781611974690.ch10. URL http://epubs.siam.org/doi/abs/10.1137/1.9781611974690.ch10.

[152] Rudolf Mathon. A note on the graph isomorphism counting problem. *Information Processing Letters*, 8(3):131–136, March 1979. doi: 10.1016/0020-0190(79)90004-8. URL https://doi.org/10.1016/0020-0190(79)90004-8.

[153] Toshiki Matsumine, Toshiaki Koike-Akino, and Ye Wang. Channel decoding with quantum approximate optimization algorithm. *arXiv:1903.02537*, 2019.

[154] Jarrod R McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. The theory of variational hybrid quantum-classical algorithms. *New Journal of Physics*, 18(2):023023, 2016.

[155] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9 (1), November 2018. doi: 10.1038/s41467-018-07090-4. URL https://doi.org/10.1038/s41467-018-07090-4.

[156] Catherine McGeoch. Performance tuning for d-wave processors. *Qubits 2018 D-Wave Users Conference*, 2018.

[157] Catherine C McGeoch. Adiabatic quantum computation and quantum annealing: Theory and practice. *Synthesis Lectures on Quantum Computing*, 5(2):1–93, 2014.

[158] Brendan D. McKay and Adolfo Piperno. Practical graph isomorphism, {II}. *Journal of Symbolic Computation*, 60(0):94 – 112, 2014. ISSN 0747-7171. doi: http://dx.doi.org/10.1016/j.jsc.2013.09.003. URL http://www.sciencedirect.com/science/article/pii/S0747717113001193.

[159] Athanasios Migdalas, Panos M Pardalos, and Peter Värbrand. *Multilevel optimization: algorithms and applications*, volume 20. Springer Science & Business Media, 2013.

[160] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceedings of the 5th ACM/Usenix Internet Measurement Conference (IMC'07)*, San Diego, CA, October 2007.

[161] Susan M. Mniszewski, Christian F. A. Negre, and Hayato Ushijima-Mwesigwa. Hybrid Quantum Computing for Graph Partitioning and Community Detection. ISTI Rapid Response 2018 Hands-on Quantum Computing, 2018.

[162] Abbe Mowshowitz. Entropy and the complexity of graphs: I. an index of the relative complexity of a graph. *The Bulletin of Mathematical Biophysics*, 30(1):175–204, March 1968. doi: 10.1007/bf02476948. URL https://doi.org/10.1007/bf02476948.

[163] Abbe Mowshowitz and Matthias Dehmer. A symmetry index for graphs. *J. Math. Biophys*, 30:533–546, 2010.

[164] Danny Munera, Daniel Diaz, Salvador Abreu, and Philippe Codognet. A parametric framework for cooperative parallel local search. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 13–24. Springer, 2014.

[165] Danny Munera, Daniel Diaz, Salvador Abreu, Francesca Rossi, Vijay A Saraswat, and Philippe Codognet. Solving hard stable matching problems via local search and cooperative parallelization. In *AAAI*, pages 1212–1218, 2015.

[166] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and linear programming. *Math. Program.*, 39:117–129, 1987.

[167] Raj Rao Nadakuditi and Mark EJ Newman. Graph spectra and the detectability of community structure in networks. *Physical Review Letters*, 108(18):188701, 2012.

[168] Ken M Nakanishi, Keisuke Fujii, and Synge Todo. Sequential minimal optimization for quantum-classical hybrid algorithms. *arXiv:1903.12166*, 2019.

[169] Giacomo Nannicini. Performance of hybrid quantum-classical variational heuristics for combinatorial optimization. *Physical Review E*, 99(1), January 2019. doi: 10.1103/physreve.99.013304. URL `https://doi.org/10.1103/physreve.99.013304`.

[170] John Napp, Rolando L. La Placa, Alexander M. Dalzell, Fernando G. S. L. Brandao, and Aram W. Harrow. Efficient classical simulation of random shallow 2d quantum circuits, 2019.

[171] Uwe Naumann and Olaf Schenk. *Combinatorial Scientific Computing*. Chapman & Hall/CRC, 1st edition, 2012. ISBN 1439827354, 9781439827352.

[172] Christian FA Negre, Hayato Ushijima-Mwesigwa, and Susan M Mniszewski. Detecting multiple communities using quantum annealing on the d-wave system. *arXiv preprint arXiv:1901.09756*, 2019.

[173] John A Nelder and Roger Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[174] NetworkX. Erdős-rényi random graph generator. URL `https://networkx.github.io/documentation/stable/reference/generated/networkx.generators.random_graphs.erdos_renyi_graph.html#networkx.generators.random_graphs.erdos_renyi_graph`.

[175] F. Neukart, G. Compostella, C. Seidel, D. von Dollen, S. Yarkoni, and B. Parney. Traffic flow optimization using a quantum annealer. *Frontiers in ICT*, 4:1–6, 2017.

[176] M. E. J. Newman. From the cover: Modularity and community structure in networks. *Proceedings of the National Academy of Science*, 103:8577–8582, 2006. doi: 10.1073/pnas.0601602103.

[177] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. In *NIPS*, volume 14, pages 849–856, 2001.

[178] Carlo Nicolini, Cécile Bordier, and Angelo Bifone. Community detection in weighted brain connectivity networks beyond the resolution limit. *Neuroimage*, 146:28–39, 2017.

[179] Sergey Novikov, Robert Hinkey, Steven Disseler, James I Basham, Tameem Albash, Andrew Risinger, David Ferguson, Daniel A Lidar, and Kenneth M Zick. Exploring more-coherent quantum annealing. *arXiv preprint arXiv:1809.04485*, 2018.

[180] Daniel O'Malley, Velimir V. Vesselinov, Boian S. Alexandrov, and Ludmil B. Alexandrov. Nonnegative/binary matrix factorization with a D-Wave quantum annealer. *PLOS ONE*, 13 (12):e0206653, 2018.

[181] Gurobi Optimization. Inc.,"gurobi optimizer reference manual," 2015. `http://www.gurobi.com`, 2014.

[182] JS Otterbach, R Manenti, N Alidoust, A Bestwick, M Block, B Bloom, S Caldwell, N Didier, E Schuyler Fried, S Hong, et al. Unsupervised machine learning on a hybrid quantum computer. *arXiv:1712.05771*, 2017.

[183] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[184] David A Papa and Igor L Markov. Hypergraph partitioning and clustering., 2007.

[185] Christos H Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of computer and system sciences*, 43(3):425–440, 1991.

[186] Ojas D. Parekh, Ciaran Ryan-Anderson, and Sevag Gharibian. Quantum optimization and approximation algorithms., 2019. URL `https://doi.org/10.2172%2F1492737`.

[187] Ferran Parés, Dario Garcia Gasulla, Armand Vilalta, Jonatan Moreno, Eduard Ayguadé, Jesús Labarta, Ulises Cortés, and Toyotaro Suzumura. Fluid communities: a competitive, scalable and diverse community detection algorithm. In *International Conference on Complex Networks and their Applications*, pages 229–240. Springer, 2017.

[188] Jongsoo Park, Mikhail Smelyanskiy, Ulrike Meier Yang, Dheevatsa Mudigere, and Pradeep Dubey. High-performance algebraic multigrid solver optimized for multi-core based distributed parallel systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 54. ACM, 2015.

[189] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[190] WangChun Peng, BaoNan Wang, Feng Hu, YunJiang Wang, XianJin Fang, XingYuan Chen, and Chao Wang. Factoring larger integers with fewer qubits via quantum annealing with optimized parameters. *SCIENCE CHINA Physics, Mechanics & Astronomy*, 62(6):60311, 2019.

[191] María Pérez-Ortiz, Pedro Antonio Gutiérrez, and César Hervás-Martínez. Projection-based ensemble learning for ordinal regression. *IEEE transactions on cybernetics*, 44(5):681–694, 2013.

[192] Jason R Petta, Alexander Comstock Johnson, Jacob M Taylor, Edward A Laird, Amir Yacoby, Mikhail D Lukin, Charles M Marcus, Micah P Hanson, and Arthur C Gossard. Coherent manipulation of coupled electron spins in semiconductor quantum dots. *Science*, 309(5744): 2180–2184, 2005.

[193] Michael J. D. Powell. The NEWUOA software for unconstrained optimization without derivatives. In *Large-scale Nonlinear Optimization*, pages 255–297. Springer, 2006.

[194] Michael JD Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer, 1994.

[195] Michael JD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages 26–46, 2009.

[196] MJD Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7:287–336, 1998.

[197] Zirou Qiu, Ruslan Shaydulin, Xiaoyuan Liu, Yuri Alexeev, Christopher S. Henry, and Ilya Safro. ELRUNA: Elimination Rule-based Network Alignment. *preprint at arXiv:1911.05486*, 2019.

[198] Rigetti. Grove, 2019. URL `https://github.com/rigetti/grove`.

[199] Giovanni Rinaldi. The Max-Cut problem: Challenges from quantum computing. Presented at the Séminaire du Group POC LIP6 de Sorbonne Universite, Oct 2018.

[200] A. H. G. Rinnooy Kan and G. T. Timmer. Stochastic global optimization methods, part I: Clustering methods. *Mathematical Programming*, 39(1):27–56, 1987. doi: 10.1007/BF02592070.

[201] A. H. G. Rinnooy Kan and G. T. Timmer. Stochastic global optimization methods, part II: Multi level methods. *Mathematical Programming*, 39(1):57–78, 1987. doi: 10.1007/BF02592071.

[202] Jonathan Romero, Ryan Babbush, Jarrod McClean, Cornelius Hempel, Peter Love, and Alán Aspuru-Guzik. Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz. *Quantum Science and Technology*, 2018.

[203] Jonathon Romero, Jonathon P. Olson, and Alan Aspuru-Guzik. Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology*, 2(4):045001, 2017.

[204] Dorit Ron, Ilya Safro, and Achi Brandt. A fast multigrid algorithm for energy minimization under planar density constraints. *Multiscale Modeling & Simulation*, 8(5):1599–1620, 2010.

[205] Dorit Ron, Ilya Safro, and Achi Brandt. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation*, 9(1):407–423, 2011.

[206] Gili Rosenberg, Mohammad Vazifeh, Brad Woods, and Eldad Haber. Building an iterative heuristic solver for a quantum annealer. *Computational Optimization and Applications*, 65(3): 845–869, Dec 2016. ISSN 1573-2894. doi: 10.1007/s10589-016-9844-y. URL `https://doi.org/10.1007/s10589-016-9844-y`.

[207] Randolf Rotta and Andreas Noack. Multilevel local search algorithms for modularity clustering. *Journal of Experimental Algorithmics*, 16:2–3, 2011.

[208] Thomas Harvey Rowan. *Functional stability analysis of numerical algorithms.* PhD thesis, University of Texas at Austin, 1990.

[209] Ehsan Sadrfaridpour, Jeereddy Sandeep, Ken Kennedy, Andre Luckow, Talayeh Razzaghi, and Ilya Safro. Algebraic multigrid support vector machines. In *ESANN 2017 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, 2017. ESANN. ISBN 9782875870391. URL `https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2017-37.pdf`.

[210] Ehsan Sadrfaridpour, Talayeh Razzaghi, and Ilya Safro. Engineering fast multilevel support vector machines. *Machine Learning*, pages 1–39, 2019.

[211] Mark Saffman, Thad G Walker, and Klaus Mølmer. Quantum information with rydberg atoms. *Reviews of Modern Physics*, 82(3):2313, 2010.

[212] Ilya Safro and Boris Temkin. Multiscale approach for the network compression-friendly ordering. *J. Discrete Algorithms*, 9(2):190–202, 2011.

[213] Ilya Safro, Dorit Ron, and Achi Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24–41, 2006.

[214] Ilya Safro, Dorit Ron, and Achi Brandt. Multilevel algorithms for linear ordering problems. *ACM Journal of Experimental Algorithmics*, 13, 2008.

[215] Ilya Safro, Peter Sanders, and Christian Schulz. Advanced coarsening schemes for graph partitioning. *ACM Journal of Experimental Algorithmics (JEA)*, 19:2–2, 2015.

[216] Pratha Sah, Lisa O Singh, Aaron Clauset, and Shweta Bansal. Exploring community structure in biological networks with random graphs. *BMC Bioinformatics*, 15(1):220, 2014.

[217] Peter Sanders and Christian Schulz. Think Locally, Act Globally: Highly Balanced Graph Partitioning. In *Proceedings of the 12th International Symposium on Experimental Algorithms (SEA'13)*, volume 7933 of *LNCS*, pages 164–175. Springer, 2013.

[218] Sebastian Schlag, Vitali Henne, Tobias Heuer, Henning Meyerhenke, Peter Sanders, and Christian Schulz. k-way hypergraph partitioning via n-level recursive bisection. In *2016 Proceedings of the Eighteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 53–67. SIAM, 2016.

[219] Bart Selman, Henry A Kautz, Bram Cohen, et al. Local search strategies for satisfiability testing. *Cliques, coloring, and satisfiability*, 26:521–532, 1993.

[220] Ruslan Shaydulin and Yuri Alexeev. Evaluating quantum approximate optimization algorithm: A case study. In *2019 Tenth International Green and Sustainable Computing Conference (IGSC)*. IEEE, October 2019. doi: 10.1109/igsc48788.2019.8957201. URL `https://doi.org/10.1109/igsc48788.2019.8957201`.

[221] Ruslan Shaydulin and Ilya Safro. Aggregative Coarsening for Multilevel Hypergraph Partitioning. In Gianlorenzo D'Angelo, editor, *17th International Symposium on Experimental Algorithms (SEA 2018)*, volume 103 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:15, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-070-5. doi: 10.4230/LIPIcs.SEA.2018.2. URL `http://drops.dagstuhl.de/opus/volltexte/2018/8937`.

[222] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. Community detection across emerging quantum architectures. *Proceedings of the 3rd International Workshop on Post Moore's Era Supercomputing*, 2018.

[223] Ruslan Shaydulin, Jie Chen, and Ilya Safro. Relaxation-based coarsening for multilevel hypergraph partitioning. *SIAM Multiscale Modeling and Simulation*, 17:482–506, 2019.

[224] Ruslan Shaydulin, Ilya Safro, and Jeffrey Larson. Multistart methods for quantum approximate optimization. *2019 IEEE High Performance Extreme Computing Conference (HPEC)*, 2019.

[225] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Christian F. A. Negre, Ilya Safro, Susan M. Mniszewski, and Yuri Alexeev. A hybrid approach for solving optimization problems on small quantum computers. *Computer*, 52(6):18–26, June 2019. doi: 10.1109/mc.2019.2908942. URL `https://doi.org/10.1109/mc.2019.2908942`.

[226] Ruslan Shaydulin, Hayato Ushijima-Mwesigwa, Ilya Safro, Susan Mniszewski, and Yuri Alexeev. Network community detection on small quantum computers. *Advanced Quantum Technologies*, 2(9):1900029, June 2019. doi: 10.1002/qute.201900029. URL `https://doi.org/10.1002/qute.201900029`.

[227] Yu-Bo Sheng and Lan Zhou. Deterministic entanglement distillation for secure double-server blind quantum computation. *Scientific reports*, 5:7815, 2015.

[228] Yu-Bo Sheng and Lan Zhou. Distributed secure quantum machine learning. *Science Bulletin*, 62(14):1025–1029, 2017.

[229] Yu-Bo Sheng and Lan Zhou. Blind quantum computation with a noise channel. *Physical Review A*, 98(5):052343, 2018.

[230] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Departmental Papers (CIS)*, page 107, 2000.

[231] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994.

[232] Gábor Simonyi. Graph entropy: a survey. *Combinatorial Optimization*, 20:399–441, 1995.

[233] A.J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. *Journal of Global Optimization*, 29(2):225–241, June 2004. doi: 10.1023/b:jogo.0000042115.44455.f3. URL `https://doi.org/10.1023/b:jogo.0000042115.44455.f3`.

[234] Christian Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. Networkit: An interactive tool suite for high-performance network analysis. *CoRR, abs/1403.3005*, 2014.

[235] Alexander Strehl and Joydeep Ghosh. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res*, 3(Dec):583–617, 2002.

[236] Gang Su, Allan Kuchinsky, John H Morris, David J States, and Fan Meng. GLay: community structure analysis of biological networks. *Bioinformatics*, 26(24):3135–3137, 2010.

[237] Liang Sun, Shuiwang Ji, and Jieping Ye. Hypergraph spectral learning for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 668–676. ACM, 2008.

[238] Mario Szegedy. What do qaoa energies reveal about graphs?, 2019.

[239] Ewin Tang. A quantum-inspired classical algorithm for recommendation systems. *arXiv preprint arXiv:1807.04271*, 2018.

[240] Ole Tange. *GNU Parallel 2018*. Ole Tange, March 2018. ISBN 9781387509881. doi: 10.5281/zenodo.1146014. URL `https://doi.org/10.5281/zenodo.1146014`.

[241] Gabriel Tavares. *New algorithms for Quadratic Unconstrained Binary Optimization (QUBO) with applications in engineering and social sciences*. PhD thesis, Rutgers University-Graduate School-New Brunswick, 2008.

[242] Jason P Terry, Prosper D Akrobotu, Christian F A Negre, and Susan M Mniszewski. Quantum isomer search. *PLoS ONE*, 15(1):e0226787, 2020.

[243] Tony T Tran, Minh Do, Eleanor G Rieffel, Jeremy Frank, Zhihui Wang, Bryan O'Gorman, Davide Venturelli, and J Christopher Beck. A hybrid quantum-classical approach to solving scheduling problems. In *Ninth Annual Symposium on Combinatorial Search*, 2016.

[244] Aleksandar Trifunovic. *Parallel algorithms for hypergraph partitioning*. University of London, 2006.

[245] Hayato Ushijima-Mwesigwa, Christian FA Negre, and Susan M Mniszewski. Graph partitioning using quantum annealing on the D-Wave system. In *Proceedings of the Second International Workshop on Post Moore's Era Supercomputing*, pages 22–29. ACM, 2017.

[246] Hayato M Ushijima-Mwesigwa, Christian F A Negre, Susan M Mniszewski, and Ilya Safro. Multilevel quantum annealing for graph partitioning. *Qubits 2018 D-Wave Users Conference*, 2018.

[247] Wim van Dam, Michele Mosca, and Umesh Vazirani. How powerful is adiabatic quantum computation? 2002. doi: 10.1109/SFCS.2001.959902.

[248] Brendan Vastenhouw and Rob H Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM Review*, 47(1):67–95, 2005.

[249] Guillaume Verdon, Michael Broughton, and Jacob Biamonte. A quantum algorithm to train neural networks using low-depth circuits. *arXiv:1712.05304*, 2017.

[250] Guillaume Verdon, Michael Broughton, Jarrod R McClean, Kevin J Sung, Ryan Babbush, Zhang Jiang, Hartmut Neven, and Masoud Mohseni. Learning to learn with quantum neural networks via classical neural networks. *arXiv preprint arXiv:1907.05415*, 2019.

[251] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[252] Stefan Voß, Silvano Martello, Ibrahim H Osman, and Catherine Roucairol. *Meta-heuristics: Advances and trends in local search paradigms for optimization*. Springer Science & Business Media, 2012.

[253] Willem Waegeman and Luc Boullart. An ensemble of weighted support vector machines for ordinal regression. *International Journal of Computer Systems Science and Engineering*, 3(1): 47–51, 2009.

[254] C. Walshaw. Multilevel refinement for combinatorial optimisation problems. *Annals Oper. Res.*, 131:325–372, 2004.

[255] Yang Wang, Zhipeng Lü, Fred Glover, and Jin-Kao Hao. Effective variable fixing and scoring strategies for binary quadratic programming. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 72–83. Springer, 2011.

[256] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97:022304, 2018.

[257] Duncan J Watts. Networks, dynamics, and the small-world phenomenon. *American Journal of Sociology*, 105(2):493–527, 1999.

[258] Dave Wecker, Matthew B Hastings, and Matthias Troyer. Training a quantum optimizer. *Physical Review A*, 94(2):022309, 2016.

[259] Max Wilson, Sam Stromswold, Filip Wudarski, Stuart Hadfield, Norm M Tubman, and Eleanor Rieffel. Optimizing quantum heuristics with meta-learning. *arXiv preprint arXiv:1908.03185*, 2019.

[260] Christopher J. Wood and John A. Smolin. A C++ quantum circuit simulator with realistic noise. `https://github.com/Qiskit/qiskit-terra/tree/master/src/qasm-simulator-cpp`. [Online; accessed September 21, 2018].

[261] Yuezhong Wu, Thomas Weise, and Raymond Chiong. Local search for the traveling salesman problem: A comparative study. In *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, pages 213–220. IEEE, 2015.

[262] L Xiao and GL Long. Fetching marked items from an unsorted database in nmr ensemble computing. *Physical Review A*, 66(5):052320, 2002.

[263] Ulrike Meier Yang et al. Boomeramg: a parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002.

[264] Zhi-Cheng Yang, Armin Rahmani, Alireza Shabani, Hartmut Neven, and Claudio Chamon. Optimizing variational quantum algorithms using Pontryagin's minimum principle. *Physical Review X*, 7(2):021027, 2017.

[265] Ehsan Zahedinejad, Daniel Crawford, Clemens Adolphs, and Jaspreet S Oberoi. Multi-community detection in signed graphs using quantum hardware. *arXiv preprint arXiv:1901.04873*, 2019.

[266] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, volume 19, pages 1633–1640, 2006.

[267] Leo Zhou, Sheng-Tao Wang, Soonwon Choi, Hannes Pichler, and Mikhail D Lukin. Quantum approximate optimization algorithm: Performance, mechanism, and implementation on near-term devices. *arXiv:1812.01041*, 2018.

[268] D Zhu, NM Linke, M Benedetti, KA Landsman, NH Nguyen, CH Alderete, A Perdomo-Ortiz, N Korda, A Garfoot, C Brecque, et al. Training of quantum circuits on a hybrid quantum computer. *arXiv:1812.08862*, 2018.

[269] Ilia Zintchenko, Matthew B. Hastings, and Matthias Troyer. From local to global ground states in ising spin glasses. *Phys. Rev. B*, 91:024201, Jan 2015. doi: 10.1103/PhysRevB.91.024201. URL `https://link.aps.org/doi/10.1103/PhysRevB.91.024201`.